FICS 2010

Dale Miller, Arnaud Carayol, Panos Rondogiannis, Lars Birkedal, Marek Czarnecki, Hervé Grall, Paul Levy, Matteo Mio, Keiko Nakata, Andrei Romashchenko, et al.

Fixed Points in Computer Science 2010

Brno, August 21-22, 2010

Preface

FICS 2010, the 7th Workshop on *Fixed Points in Computer Science*, was held in Brno, Czech Republic, on August 21-22 2010, as a satellite workshop to the conferences *Mathematical Foundations of Computer Science* and *Computer Science Logic* 2010. FICS aims to provide a forum for researchers to present their results to those members of the computer science and logic communities who study or apply the theory of fixed points.

The editor wishes to thank the authors that primarily contributed with their ideas and work to the success of the workshop. He would also like to thank the the external reviewers, Dietmar Berwanger, Martín Escardó, Martin Lange, Richard McKinley, Dieter Probst, Joshua Sack, and the members of the Program Committee, Thorsten Altenkirch, Giovanna d'Agostino, Peter Dybjer, Zoltán Ésik, Anna Ingólfsdóttir, Gerhard Jäger, Ralph Matthes, Andrzej Murawski, Damian Niwinski, Luigi Santocanale, Alex Simpson, Jean-Marc Talbot, Tarmo Uustalu, Yde Venema, Igor Walukiewicz, for their collaboration in the process of selecting the contributions. Finally, the editor wishes to thank Tony Kučera and Tomáš Brázdil for taking care of the local organization of the workshop.

A special thank goes to the French institutions that made the organisation of the workshop possible through their financial support:

- Laboratoire d'Informatique Fondamentale de Marseille (LIF, UMR 6166),
- Université Aix-Marseille I (Université de Provence),
- Agence Nationale de la Recherche,
  through the projects ECSPER (ANR-JC09-472677) and SFINCS (ANR- 07-SESU-012).

Brno, August 21, 2010,
Luigi Santocanale

Table of Contents

Invited Talks

Contributed Talks

Structures defined by higher-order recursion schemes

Arnaud Carayol
LIGM
Université Paris-Est and CNRS
5, boulevard Descartes, Champs-sur-Marne
F-77454 Marne-la-Vallée Cedex 2
carayol@univ-mlv.fr

Higher-order recursion schemes are a particular form of typed term rewriting systems. By iterative rewriting from a fixed finite term, an (higher-order) recursion scheme generates at the limit an infinite term. In 2006, Luke Ong has shown that for the infinite terms obtained in this way monadic second-order logic (MSO) is decidable. As terms defined by recursion schemes encompass most of the known examples of infinite terms having decidable MSO theory, this result has revived the interest for recursion schemes.

In this talk, we will survey results concerning the structures defined by recursion schemes. These structures are essentially graphs which can be defined using MSO logic on the terms generated by recursion schemes. Our main focus will be to assess the expressivity of these structures through alternative characterisations.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 7-7

Fixed points and proof theory: An extended abstract

Dale Miller
INRIA Saclay & LIX/Ecole Polytechnique, Palaiseau, France

Abstract

We overview some recent results in the proof theory for fixed points and illustrate how those results can be used to provide a unified approach to computation, model checking, and theorem proving. A key theme in this work is the development of *focused proof system* that allow for micro-rules (*e.g.*, sequent calculus introduction rules) to be organized into macro-rules. Such macro-rules can often be designed to correspond closely to "steps" or "actions" taken within computational systems.

1 Introduction

Potentially unbounded behavior is available in proof theory via the contraction rule (see Figure 1): when reading this rule from its conclusion to its premise, a formula is duplicated and such duplication can be applied again and again. When contraction is removed, decidability often follows. For example, the multiplicative-additive fragment of linear logic (MALL) does not have contraction (nor weakening) and, as a result, it is decidable whether or not a formula is a theorem. It is possible to given a proof system for propositional intuitionistic logic that does not contain contraction [9]: a naive implementation of that proof system can be a decision procedure for that logic.

Decidable logics, such as MALL and propositional intuitionistic logic, have rather limited applications. For a logic and proof system to find wide ranging applications in, say, computation and deduction, they must allow for potentially infinite behaviors.

Girard designed *linear logic* to be the result of extending MALL with the exponentials (!, ?): the proof rules of contraction and weakening are allowed only on formulas marked by such exponentials. In this way, potentially infinite behaviors can be mixed with MALL: linear logic becomes undecidable in this way (even the propositional fragment of linear logic is undecidable [12]).

While the addition of the exponentials to MALL is elegant and powerful, the exponentials are not without problems for those working in computational logic. For example, when using (as we will) *focused proof systems*, the exponentials can cause focusing phases to terminate. Furthermore, exponentials are not canonical: that is, if one allows a "red" version and a "blue" version of both of ! and ?, it is not possible to prove them equivalent (all other connectives of linear logic are canonical in this sense). While there are interesting uses for such non-canonical exponentials (see, for example, the material on *subexponentials* in [18]) one would like to have alternative approaches to potentially infinite behavior in proof systems.

Baelde and Miller have proposed another approach to extending MALL with potentially infinite behavior: in particular, they have added the least and greatest fixed points directly to MALL [6, 3]. The resulting logic is surprisingly elegant and expressive: it has been used to describe the logical foundations of a model checker [5] and is being used to design a new theorem proving architecture for the search for proofs involving induction [7]. The development of a *focused proof system* for MALL plus fixed points has lead to some new proof theory phenomena as well as some new ways to approach the unity of such different topics as computation, model checking, and inductive and co-inductive theorem proving.

This extended abstract will illustrate some of the proof theory of fixed points by concentrating on adding fixed points to classical first-order logic. Parts of this abstract are based on similar material found in [14].

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 9-16

$$\text{Structural rules:}\quad \frac{\vdash B, B, \Delta}{\vdash B, \Delta}\ contract \qquad \frac{\vdash \Delta}{\vdash B, \Delta}\ weaken$$

$$\text{Identity rules:}\quad \frac{}{\vdash B, B^{\perp}}\ initial \qquad \frac{\vdash \Delta_1, B \quad \vdash B^{\perp}, \Delta_2}{\vdash \Delta_1, \Delta_2}\ cut$$

$$\text{Multiplicative rules:}\quad \frac{\vdash B_1, \Delta_1 \quad \vdash B_2, \Delta_2}{\vdash B_1 \wedge B_2, \Delta_1, \Delta_2}\ \wedge \qquad \frac{\vdash B_1, B_2, \Delta}{\vdash B_1 \vee B_2, \Delta}\ \vee$$

$$\frac{}{\vdash t}\ t \qquad \frac{\vdash \Delta}{\vdash f, \Delta}\ f$$

$$\text{Additive rules:}\quad \frac{\vdash B_1, \Delta \quad \vdash B_2, \Delta}{\vdash B_1 \wedge B_2, \Delta}\ \wedge \qquad \frac{\vdash B_i, \Delta}{\vdash B_1 \vee B_2, \Delta}\ \vee_i$$

$$\frac{}{\vdash t, \Delta}\ t \qquad \text{—}$$

Figure 1: Proof rules for propositional classical logic

2 A focused proof system for classical logic

We now consider propositional classical logic over formulas built from $\wedge$, $\vee$, t, f. Remarkably, we shall not have a need for atomic formulas. Figure 1 contains a one-sided sequent calculus proof systems for this logic. A sequent $\vdash \Delta$ contains a *multiset* of formulas Δ. Besides containing the usual structural rules and identity rules (initial and cut), the logical connectives have *two* sets of introduction rules. That is, for each logical connective, there is a set of *multiplicative* rules and a set of *additive* rules for it. (The set of additive rules for f is empty and the set of additive rules for $\vee$ contains two rules.) Notice that the rules for the additive conjunction and the multiplicative disjunction are *invertible* while the rules for the multiplicative conjunction and the additive disjunction are not necessarily invertible.

The cut rule is admissible and the initial rule is admissible for all but atomic formulas. Since we have no atomic formulas, both identity rules can be dropped.

As is well known, the additive rule and the multiplicative rule for the same connective are inter-admissible in the presence of the structural rules of contraction and weakening. Another way to state this admissibility statement is the following. Annotate names of the connectives in the inference rules as follows: the invertible rules are annotated as negative $\wedge^-$ and $\vee^-$ while the not-necessarily-invertible rules are annotated as positive $\wedge^+$ and $\vee^+$ (annotate their units also to be t^-, f^-, t^+, f^+). Given a formula B let $\hat{B}$ be any *polarization* of B in which every logical connectives in B is given either a plus or a minus annotation. Then, B is provable (in the unannotated proof system) if and only if $\hat{B}$ is provable (in the annotated proof system). We shall say that an annotated formula is *negative* if its top-level logical connective is annotated negatively and is *positive* if its top-level logical connective is annotated positively. By $B^{\perp}$ we shall mean the negation normal form of B: since we have no atomic formulas, negations will not occur within formulas. Notice that the de Morgan dual of $\vee^+$ and $\wedge^+$ are, respectively, $\wedge^-$ and $\vee^-$.

If we try to take the construction of proofs literally as a model for performing computation, one is immediately struck by the inappropriateness of sequent calculus rules in Figure 1 for this task: there are just too many ways to build proofs and most of them differ in truly minor ways. One would like to have a tight correspondence between the application of inference rules and "actions" within a computation. An early attempt to provide the sequent calculus with normal forms that corresponded to computation was the work on *uniform proofs* and *backchaining* [15] that provided a proof-theoretic foundation for logic programming. It was, however, Andreoli's *focused proof system* [1] for linear logic that really allowed one to more richly restrict and organize the sequent calculus. The earlier work on uniform proofs was

$$\text{Structural rules:}\quad \frac{\vdash \Theta, P \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, P}\ \textit{Store} \qquad \frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\ \textit{Release} \qquad \frac{\vdash P, \Theta \Downarrow P}{\vdash P, \Theta \Uparrow \cdot}\ \textit{Focus}$$

$$\text{Negative phase:}\quad \frac{}{\vdash \Theta \Uparrow \Gamma, t^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, A \quad \vdash \Theta \Uparrow \Gamma, B}{\vdash \Theta \Uparrow \Gamma, A \wedge^- B}$$

$$\frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow \Gamma, f^-} \qquad \frac{\vdash \Theta \Uparrow \Gamma, A, B}{\vdash \Theta \Uparrow \Gamma, A \vee^- B} \qquad \frac{\vdash \Theta \Uparrow \Gamma, A}{\vdash \Theta \Uparrow \Gamma, \forall x A}$$

$$\text{Positive phase:}\quad \frac{}{\vdash \Theta \Downarrow t^+} \qquad \frac{\vdash \Theta \Downarrow A \quad \vdash \Theta \Downarrow B}{\vdash \Theta \Downarrow A \wedge^+ B}$$

$$\frac{\vdash \Theta \Downarrow A_i}{\vdash \Theta \Downarrow A_1 \vee^+ A_2} \qquad \frac{\vdash \Theta \Downarrow A[t/x]}{\vdash \Theta \Downarrow \exists x A}$$

Figure 2: The focused proof system LKF for classical logic.

rather limited, both in its range of "focusing behavior" and in the subsets of logic to which it could be applied. Andreoli's result, however, applied to a full and rich logic.

Figure 2 presents the focused proof system for *annotated* propositional classical logic that is derived from the LKF proof system of Liang and Miller [11]. In that figure, P denotes a positive formula, N a negative formula, and Θ consists of positive formulas. The sequents in this proof system are of the form $\vdash \Theta \Uparrow \Gamma$ and $\vdash \Theta \Downarrow B$ where Θ is a multiset of formulas, Γ is a list of formulas, and B is a formula. The inference rules used in the negative phase are invertible while those in the positive phase are not necessarily invertible. The structural rules contain two rules that allow switching between the two kinds of sequents. Notice also that the *Focus* rule incorporates the contraction rule.

Theorem 1. *LKF is sound and complete for classical logic. More precisely, let B be a first order formula and let $\hat{B}$ be a polarization of B. Then B is provable in classical logic if and only if there is an LKF proof of $\vdash \cdot \Uparrow \hat{B}$ [11].*

Notice that polarization does not affect provability but it does affect the shape of possible LKF proofs.

Focused proof systems such as LKF allow us to change the size of inference rules with which we work. Let us call individual introduction rules (such as in Figures 1 and 2) "micro-rules." An entire phase within a focused proof (the collecting together of only $\Uparrow$ sequents or only $\Downarrow$ sequent) can be seen as a "macro-rule." In particular, consider the following derivation

$$\cfrac{\cfrac{\vdash \Theta, D \Uparrow N_1 \quad \cdots \quad \vdash \Theta, D \Uparrow N_n}{\cdots \text{ only } \Downarrow \text{ sequents here } \cdots}}{\cfrac{\vdash \Theta, D \Downarrow D}{\vdash \Theta, D \Uparrow \cdot}}$$

Here, the selection of the formula D for the focus can be taken as selecting among several macro-rules: this derivation illustrates one such macro-rule: the inference rule with conclusion $\vdash \Theta, D \Uparrow \cdot$ and with $n \geq 0$ premises $\vdash \Theta, D \Uparrow N_1, \ldots, \vdash \Theta, D \Uparrow N_n$ (where $N_1, \ldots, N_n$ are negative formulas). We shall say that this macro-rule is positive. Similarly, there is a corresponding negative macro-rule with conclusion, say, $\vdash \Theta, D \Uparrow N_i$, and with $m \geq 0$ premises of the form $\vdash \Theta, D, \mathscr{C} \Uparrow \cdot$, where $\mathscr{C}$ is a multiset of positive formulas or negative literals. In this way, focused proofs allow us to view the construction of proofs from conclusions of the form $\vdash \Theta \Uparrow \cdot$ by first attaching a positive macro rule (by focusing on some formula in Θ) and then attaching negative inference rules to the resulting premises until one arrives again to sequents of the form $\vdash \Theta' \Uparrow \cdot$. Such a combination of a positive macro rule below negative macro rules is often called a *bipole* [2].

$$\text{Quantifiers:}\quad \frac{\vdash B[t/x],\Delta}{\vdash \exists x.B,\Delta} \qquad \frac{\vdash B[y/x],\Delta}{\vdash \forall x.B,\Delta}$$

$$\text{Fixed points:}\quad \frac{\vdash B(\nu B)\bar{u},\Delta}{\vdash \nu B\bar{u},\Delta} \qquad \frac{\vdash B(\mu B)\bar{u},\Delta}{\vdash \mu B\bar{u},\Delta}$$

$$\text{Equality:}\quad \frac{\vdash \Delta\sigma}{\vdash u\neq v,\Delta}\ \dagger \qquad \frac{}{\vdash u\neq v,\Delta}\ \ddagger \qquad \frac{}{\vdash u=u,\Delta}$$

The † proviso requires the terms u and v to be unifiable and σ to be their most general unifier. The ‡ proviso requires that the terms s and t are not unifiable. The usual eigenvariable condition holds for y in the $\forall$ introduction rule.

Figure 3: Introduction rules for $=$ and μ.

3 Fixed points and first-order structure

To classical logic we shall now add the two fixed point constructors μ and ν, the two first-order quantifiers $\forall$ and $\exists$, and the two comparison relations on first-order terms $=$ and $\neq$. Notice that each pair of these connectives forms a de Morgan dual. The connectives μ and ν are really a collection $\{\mu_n\}_{n\geq 0}$ and $\{\nu_n\}_{n\geq 0}$ such that the simple type of μ_n and of ν_n is $\tau_n \to \tau_n$, where τ_n is $i \to \ldots \to i \to o$ (n occurrences of the type i). We shall not write the subscripted arity. Notice that all six of these constants are *logical connectives* in the sense that they all have introduction rules: furthermore, the identity rules (cut and initial) are admissible. The (unfocused) proof rules for these additional connectives are given in Figure 3. The rules for equality and its negation are due to Schroeder-Heister [20] and Girard [10].

Example 2. *Identify the natural numbers as terms involving* 0 *for zero and s for successor. The following simple logic program defines two predicates on natural numbers.*

$$\begin{array}{rcl} nat\ 0 & \subset & true. \\ nat\ (s\ X) & \subset & nat\ X. \\ leq\ 0\ Y & \subset & true. \\ leq\ (s\ X)\ (s\ Y) & \subset & leq\ X\ Y. \end{array}$$

The predicate nat can be written as the fixed point

$$\mu(\lambda p\lambda x.(x=0) \vee^{+} \exists y.(s\ y) = x \wedge^{+} p\ y)$$

and binary predicate leq (less-than-or-equal) can be written as the fixed point

$$\mu(\lambda q\lambda x\lambda y.(x=0) \vee^{+} \exists u\exists v.(s\ u) = x \wedge^{+} (s\ v) = y \wedge^{+} q\ u\ v).$$

In a similar fashion, any Horn clause specification can be made into fixed point specifications (mutual recursions requires standard encoding techniques). Notice the use of positively annotated logical connectives here.

Following the usual convention of classifying invertible introduction as *negative* connectives, $\forall$ and $\neq$ are negative and their duals $\exists$ and $=$ are positive. Since the two fixed points are (currently) indistinguishable (they have the same introduction rules), we arbitrarily classify μ as positive and ν as negative. The focused versions of the introduction rules for these connectives is given in Figure 4. Notice now that focusing phases can involve long chains of unfoldings. For example, the fixed points in Example 2 are encoded using *entirely positive* connectives. Thus, if the focused sequent $\vdash \Theta \Downarrow leq\ m\ n$ has a proof, it has a proof that involves exactly one macro-rule. As the following example illustrates, it is now possible to put an arbitrary (Horn clause) computation within a (macro) rule.

$$\frac{\vdash \Theta \Downarrow B[t/x]}{\vdash \Theta \Downarrow \exists x.B} \qquad \frac{\vdash \Theta \Uparrow \Gamma, B[y/x]}{\vdash \Theta \Uparrow \Gamma, \forall x.B}$$

$$\frac{}{\vdash \Theta \Downarrow u = u} \qquad \frac{\vdash \Theta\sigma \Uparrow \Gamma\sigma}{\vdash \Theta \Uparrow \Gamma, u \neq v}\ \dagger \qquad \frac{}{\vdash \Theta \Uparrow \Gamma, u \neq v}\ \ddagger$$

$$\frac{\vdash \Theta \Uparrow \Gamma, B(\nu B)\bar{u}}{\vdash \Theta \Uparrow \Gamma, \nu B\bar{u}} \qquad \frac{\vdash \Theta \Downarrow B(\mu B)\bar{u}}{\vdash \Theta \Downarrow \mu B\bar{u}}$$

Figure 4: Focused inference rules for $=$ and μ. The proviso $\dagger$ and $\ddagger$ and the definition of σ are the same as above.

Example 3. *Consider proving the positive focused sequent*

$$\vdash \Theta \Downarrow (leq\ m\ n \wedge^+ N_1) \vee^+ (leq\ n\ m \wedge^+ N_2),$$

where m and n are natural numbers and leq is the fixed point expression displayed above. If both N_1 and N_2 are negative formulas, then there are exactly two possible macro rules with this conclusion: one with premise $\vdash \Theta \Uparrow N_1$ when $m \leq n$ and one with premise $\vdash \Theta \Uparrow N_2$ when $n \leq m$ (thus, if $m = n$, either premise is possible). In this sense, a macro inference rule can contain an entire Prolog-style computation.

The following example illustrates that fixed points can be used to describe a typical query from model checking.

Example 4. *Assume that first-order terms encode process and action expressions such as are found in CCS. The usual notion of one-step transition can generally be written using a simple, recursive Horn clause program: hence, the ternary relation $P \xrightarrow{A} P'$ can be encoded using a purely positive formula. The notion of simulation is then the (greatest) fixed point of the equivalence*

$$sim\ P\ Q \equiv \forall P' \forall A[P \xrightarrow{A} P' \supset \exists Q'[Q \xrightarrow{A} Q' \wedge sim\ P'\ Q']].$$

We can then chose to write the sim relation as the expression

$$\nu \lambda s \lambda P \lambda Q. \forall P' \forall A[P \xrightarrow{A} P' \supset \exists Q'[Q \xrightarrow{A} Q' \wedge s\ P'\ Q']]$$

(Here, we are assuming that the implication $B \supset C$ is rendered as $B^\perp \vee^- C$ in the polarized setting.) Although the body of this definition looks complex, it is, in fact, composed of exactly two "macro connectives" (a bipole). The expression $\forall P' \forall A[P \xrightarrow{A} P' \supset \cdot]$ is a purely negative formula and its "introduction" is invertible: since all possible actions A and continuations P' must be computed, there are no choices to be made in building a proof for this expression. On the other hand, focusing on the expression $\exists Q'[Q \xrightarrow{A} Q' \wedge^+ \cdot]$ yields a non-invertible, positive macro rule. In this way, the focused proof system is aligned directly with the structure of the actual (model-checking) problem.

The fact that an entire computation can fit within a macro rule (using purely positive fixed point expressions) provides a great deal of flexibility in designing inference rules. Such flexibility allows inference rules to be designed so that they correspond to an "action" within a given computational system. If one takes seriously using only macro rules (and hiding the details of the micro rules) then placing arbitrary computation within a macro inference rule is probably too much: the term "inference rule" is usually reserved for *decidable* steps. Thus, some care should be exercised in balancing the complexity of a macro rule with the needs of proof systems to have their correctness be decidable.

$$\frac{\vdash \Gamma, B(\mu B)\bar{u}}{\vdash \Gamma, \mu B\bar{u}}\ \mu \qquad \frac{\vdash \Gamma, S\bar{u} \quad \vdash BSx, (Sx)^{\perp}}{\vdash \Gamma, \nu B\bar{u}}\ \nu \qquad \frac{}{\vdash \mu B\bar{u}, \nu\bar{B}\bar{u}}\ \mu\nu$$

The variable S denotes a closed term of type τ_n, where the arity of ν is n. Also, x is an eigenvariable of the proof.

Figure 5: Inference rules for induction and co-induction.

Large macro rules can easily be broken up, if desired, by the use of *delays*. Within LKF, we can define the delaying operators

$$\partial^+(B) = B \wedge^+ t^+ \quad \text{and} \quad \partial^-(B) = B \wedge^- t^-.$$

Clearly, B, $\partial^-(B)$, and $\partial^+(B)$ are all logically equivalent but $\partial^-(B)$ is always negative and $\partial^+(B)$ is always positive. If one wishes to break a positive macro rule resulting from focusing on a given positive formula into smaller pieces, then one can insert $\partial^-(\cdot)$ into that formula. Similarly, inserting $\partial^+(\cdot)$ can limit the size of a negative macro rule. By inserting many delay operators, a focused proof can be made to emulate an unfocused proof [11].

4 Induction and co-induction rules

Treating fixed points simply by unfolding is, of course, limited. For example, while it is easy to define fixed points that describe the notions of "natural number", "even", and "odd", it is not possible to prove that every natural number is either even or odd. While one can simply write the appropriate formula stating that theorem, attempts at proofs of it will lead only to potentially infinite unfoldings. What is missing, of course, is induction. To this end, consider the three inference rules displayed in Figure 5 (taken from [3, 6]). Notice that the $\mu\nu$ rule is the only form of the initial rule that we shall need in this proof system. Here, the *negation* $\overline{B}$ of a body B is defined as $\lambda p.\lambda\vec{x}.(B(\lambda\vec{x}.(p\vec{x})^{\perp})\vec{x})^{\perp}$. A body B is said to be *monotonic* when for any variables p and $\bar{t}$, the negation normal and λ-normal form of $Bp\bar{t}$ does not contain any negated instance of p. We shall assume that *all bodies are monotonic*.

Notice that the rule called ν is the rule for co-induction. If we write sequents as two sided, then moving this rule to the left-side as an introduction rule for μ yields the usual induction rule. Thus, induction and co-induction are treated as perfect duals of each other.

Proposition 5. *The following inference rules are derivable:*

$$\frac{}{\vdash P, P^{\perp}}\ \mathit{init} \qquad \frac{\vdash \Gamma, B(\nu B)\bar{u}}{\vdash \Gamma, \nu B\bar{u}}\ \nu R$$

These results are standard, cf. [21]. The proof of the second one relies on monotonicity and is obtained by applying the ν rule with $B(\nu B)$ as the co-invariant. As a result of this proposition, we can replace the inference rules for μ and ν given in Figure 3 with the rules in Figure 5. It is also the case that μ and ν are now canonical. Such canonicity is gained, however, by the rather "high-price" of using an inference rule (the ν rule) that involves a higher-order variable (the invariant).

Baelde [3, 4] has proved the (relative) completeness of the focused inference rules in Figure 6 in the context of MALL: his proof involves some rather complex arguments for treating the higher-order nature of the induction/co-induction rule. That proof should, however, lift to the setting described here where classical logic replaces MALL.

Notice that in the $\Uparrow$ phase, one chooses between doing (co)induction on $\nu B\bar{t}$ and *freezing* the fixed point expression by moving it to the left of the $\Uparrow$. Since such a ν-expression is assigned a negative

$$\frac{\vdash \Gamma \Uparrow S\bar{t}, \Delta \quad \vdash \Uparrow BS\vec{x}, S\vec{x}^{\perp}}{\vdash \Gamma \Uparrow \nu B\bar{t}, \Delta}\ \vec{x}\text{ new} \qquad \frac{\vdash \Gamma, \nu B\bar{t} \Uparrow \Delta}{\vdash \Gamma \Uparrow \nu B\bar{t}, \Delta}$$

$$\frac{\vdash \Gamma \Downarrow B(\mu B)\vec{x}}{\vdash \Gamma \Downarrow \mu B\vec{x}} \qquad \frac{}{\vdash \nu \bar{B}\vec{x} \Downarrow \mu B\vec{x}}$$

Figure 6: Focused proof rules for the fixed point connectives

polarity, the *Focus* rule can never move it from the left of the arrow to the right. Thus, a ν-expression on the left of the $\Uparrow$ arrow can only be used to match its dual of the right $\Downarrow$ in the (focused version of the) $\mu\nu$ rule. For more about these rules for (co)induction, see the Baelde's PhD thesis [3] or the theorem prover based on that work [7].

5 The unity of computational logic

The proof theory of fixed points offers us a framework for comparing and combining computation, model checking, and theorem proving. The differences between these three activities can be characterized by their different uses of fixed points. Logic programming involves *may* behavior only, which involves unfolding fixed points and non-deterministically picking a path to a success. On the other hand, both model checking and theorem proving deal with *must* as well as *may* behavior. These two differ in that model checkers generally assume finite fixed points (or have specialized methods for handling loops) while (inductive) theorem provers use invariants to characterize possible infinite unfoldings. Given these rough descriptions, it is possible to see rich ways that these activities can fit together into one system (and one logic!) and enhance each other: for example, a theorem prover might prove certain symmetry lemmas (via induction) and these could be used in the model checker to reduce search space. Similarly, tabling within model checkers can be seen as lemma generation in theorem provers [16]. Also, as we have seen, logic programming-style computation can be inserted within the inference rules used in both model checking and theorem proving.

6 Related work

The proof theory of fixed points described here is largely taken from the recent papers of Baelde [3, 7]. While Baelde largely considered adding fixed point to MALL, he has also shown how that combination directly treats important subsets of intuitionistic logic. Miller and Saurin [17] have also considered fixed points using dialog games which are known to be closely related to focused proof systems. This work builds on earlier work by McDowell, Miller, and Tiu [13, 21]. The work on cyclic proofs by, say, Santocanale [19] and Brotherston and Simpson [8], is a related but alternative approach to the treatment of induction within proofs.

Acknowledgments: I wish to thank David Baelde and the anonymous reviewers of this abstract for their comments.

References

[1] Jean-Marc Andreoli. Logic programming with focusing proofs in linear logic. *J. of Logic and Computation*, 2(3):297–347, 1992.

[2] Jean-Marc Andreoli. Focussing and proof construction. *Annals of Pure and Applied Logic*, 107(1):131–163, 2001.

[3] David Baelde. *A linear approach to the proof-theory of least and greatest fixed points*. PhD thesis, Ecole Polytechnique, December 2008.

[4] David Baelde. Least and greatest fixed points in linear logic. Submitted, October 2009.

[5] David Baelde, Andrew Gacek, Dale Miller, Gopalan Nadathur, and Alwen Tiu. The Bedwyr system for model checking over syntactic expressions. In Frank Pfenning, editor, *21th Conference on Automated Deduction (CADE)*, number 4603 in LNAI, pages 391–397. Springer, 2007.

[6] David Baelde and Dale Miller. Least and greatest fixed points in linear logic. In N. Dershowitz and A. Voronkov, editors, *International Conference on Logic for Programming and Automated Reasoning (LPAR)*, volume 4790 of *LNCS*, pages 92–106, 2007.

[7] David Baelde, Dale Miller, and Zachary Snow. Focused inductive theorem proving. Accepted to IJCAR 2010., 2010.

[8] James Brotherston and Alex Simpson. Complete sequent calculi for induction and infinite descent. In *22th Symp. on Logic in Computer Science*, pages 51–62, 2007.

[9] Roy Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *J. of Symbolic Logic*, 57(3):795–807, September 1992.

[10] Jean-Yves Girard. A fixpoint theorem in linear logic. A posting to linear@cs.stanford.edu, February 1992.

[11] Chuck Liang and Dale Miller. Focusing and polarization in linear, intuitionistic, and classical logics. *Theoretical Computer Science*, 410(46):4747–4768, 2009.

[12] P. Lincoln, J. Mitchell, A. Scedrov, and N. Shankar. Decision problems for propositional linear logic. *Annals Pure Applied Logic*, 56:239–311, 1992.

[13] Raymond McDowell and Dale Miller. Reasoning with higher-order abstract syntax in a logical framework. *ACM Trans. on Computational Logic*, 3(1):80–136, 2002.

[14] Dale Miller. Finding unity in computational logic. In *ACM-BCS-Visions Conference*, April 2010.

[15] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[16] Dale Miller and Vivek Nigam. Incorporating tables into proofs. In J. Duparc and T. A. Henzinger, editors, *CSL 2007: Computer Science Logic*, volume 4646 of *LNCS*, pages 466–480. Springer, 2007.

[17] Dale Miller and Alexis Saurin. A game semantics for proof search: Preliminary results. In *Proceedings of the Mathematical Foundations of Programming Semantics (MFPS05)*, number 155 in Electronic Notes in Theoretical Computer Science, pages 543–563, 2006.

[18] Vivek Nigam and Dale Miller. Algorithmic specifications in linear logic with subexponentials. In *ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP)*, pages 129–140, 2009.

[19] Luigi Santocanale. A calculus of circular proofs and its categorical semantics. In *Proceedings of Foundations of Software Science and Computation Structures (FOSSACS02)*, number 2303 in LNCS, pages 357–371. Springer-Verlag, January 2002.

[20] Peter Schroeder-Heister. Rules of definitional reflection. In M. Vardi, editor, *Eighth Annual Symposium on Logic in Computer Science*, pages 222–232. IEEE Computer Society Press, IEEE, June 1993.

[21] Alwen Tiu. *A Logical Framework for Reasoning about Logical Specifications*. PhD thesis, Pennsylvania State University, May 2004.

Fixed-Point Semantics for Non-Monotonic Formalisms

Panos Rondogiannis
Department of Informatics & Telecommunications, University of Athens, Greece
prondo@di.uoa.gr

We examine two successful application domains for fixed-point theory, namely (non-monotonic) logic programming and Boolean grammars. We demonstrate that these two areas are quite close, and as a result interesting ideas from one area appear to have interesting counterparts in the other one.

Negation in Logic Programming: We present a recent [RW05] purely model-theoretic characterization of the semantics of logic programs with negation. The new semantics provides a logical basis for the classical well-founded semantics of logic programs [vGRS91]. The proposed approach uses a truth domain that has an uncountable linearly ordered set of truth values between *False* (the minimum element) and *True* (the maximum), with a *Zero* element in the middle. Under the new semantics, every logic program with negation has a unique *minimum* model which can be constructed as the least fixed point of the immediate consequence operator of the program. The main benefits of the new construction is that it provides a purely logical characterization of negation-as-failure and a better understanding of the fixed-point semantics of non-monotonic logic programming.

Boolean Grammars: These new grammars [Okh04] extend the context-free ones by allowing conjunction and negation in the right hand sides of rules. It has been demonstrated that Boolean grammars can be parsed efficiently and that they can express interesting languages that are not context-free. Despite their simplicity, Boolean grammars proved to be non-trivial from a semantic point of view. In particular, the use of negation makes it difficult to define a simple derivation-style semantics (such as the familiar one for context-free languages). We present the *well-founded construction* for Boolean grammars [KNR09], which provides the first general solution to the problem of assigning a proper meaning to every Boolean grammar. Our results indicate that many-valued formal language theory is the appropriate mathematical machinery underlying Boolean grammars.

References

[KNR09] Kountouriotis, V., Nomikos, Ch., Rondogiannis, P.: Well-Founded Semantics for Boolean Grammars. Information and Computation. **207(9)** (2009) 945–967.

[Okh04] Okhotin, A.: Boolean Grammars. Information and Computation. **194(1)** (2004) 19–48.

[RW05] Rondogiannis, P., Wadge, W., W.: Minimum Model Semantics for Logic Programs with Negation-as-Failure. ACM Transactions on Computational Logic. **6(2)** (2005) 441–467.

[vGRS91] van Gelder, A, Ross, K., A., Schlipf, J., S.: The Well-Founded Semantics for General Logic Programs. Journal of the ACM. **38(3)** (1991) 620–650.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 17-17

A Metric Model of Lambda Calculus with Guarded Recursion

Lars Birkedal
IT University of Copenhagen
birkedal@itu.dk

Jan Schwinghammer
Saarland University
jan@ps.uni-saarland.de

Kristian Støvring
IT University of Copenhagen
kss@itu.dk

Abstract

We give a model for Nakano's typed lambda calculus with guarded recursive definitions in a category of metric spaces. By proving a computational adequacy result that relates the interpretation with the operational semantics, we show that the model can be used to reason about contextual equivalence.

1 Introduction

Recent work in semantics of programming languages and program logics has suggested that there might be a connection between metric spaces and Nakano's typed lambda calculus with guarded recursive definitions [8]. In this paper we show that is indeed the case by developing a model of Nakano's calculus in a category of metric spaces and by showing that the model is computationally adequate. The latter is done with the use of a logical relation, whose existence is non-trivial because of the presence of recursive types. We define the relation in a novel indexed way, inspired by step-indexed models for operational semantics [2].

For space reasons, we can only briefly hint at some of the motivating applications in semantics:

In [4], Birkedal et al. gave a model of a programming language with recursive types, impredicative polymorphism and general ML-style references. The model is a Kripke model, where the set of worlds is a recursively defined metric space (with the recursion variable in negative position). A simplified variant of the recursive equation can be formulated in a version of Nakano's calculus.

Pottier [9] has recently suggested to use Nakano's calculus as a calculus of kinds in an extension of F_ω with recursive kinds, which can serve as a target calculus for a store-passing translation of a language with ML-style references. The model we present here can be extended to a model of F_ω with recursive kinds by indexing complete uniform per's over metric spaces.

Hinze [6] has shown, in the context of Haskell streams, how the uniqueness of fixed points of contractive functions in an operational setting can be used for verification. Nakano's calculus can be used to formalize such arguments, and in our model unique fixed points for contractive functions do exist.

For reasoning about imperative interactive programs, Krishnaswami et al. [7] related an efficient imperative implementation to a simple model of functional reactive programming. In current unpublished work by Krishnaswami on extending this to higher-order programs, metric spaces are being used to model the functional reactive programs to capture that all uses of recursion are guarded. A logical relation is used to relate the functional reactive programs and the imperative implementation thereof; that logical relation is defined using ideas that seem similar to those we are using in our logical relation for the adequacy proof in Section 3.

In the remainder of the paper, we present (our version of) Nakano's calculus (Section 2), define the denotational semantics and prove that it is computationally adequate (Section 3), briefly discuss some closely related work (Section 4), and conclude with some comments about future work.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 19-25

$$\frac{}{\Gamma, x{:}\tau \vdash x : \tau} \qquad \frac{}{\Gamma \vdash \underline{n} : \mathit{Int}} \qquad \frac{\Gamma, x{:}\tau \vdash t : \sigma}{\Gamma \vdash \lambda x{:}\tau.t : \tau \to \sigma} \qquad \frac{\Gamma \vdash t_1 : \bullet^n(\tau \to \sigma) \qquad \Gamma \vdash t_2 : \bullet^n \tau}{\Gamma \vdash t_1\, t_2 : \bullet^n \sigma}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \qquad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash (t_1, t_2) : \tau_1 \times \tau_2} \qquad \frac{\Gamma \vdash t : \bullet^n(\tau_1 \times \tau_2)}{\Gamma \vdash \mathit{proj}_i(t) : \bullet^n \tau_i} \qquad \frac{\bullet\Gamma \vdash t : \bullet\tau}{\Gamma \vdash t : \tau} \qquad \frac{\Gamma \vdash t : \bullet\tau_{i,j}}{\Gamma \vdash \mathit{In}_{i,j}(t) : ty_i}$$

$$\frac{\Gamma \vdash t : ty_i \qquad \Gamma, x_1{:}\bullet\tau_{i,1} \vdash t_1 : \tau \qquad \ldots \qquad \Gamma, x_{i_k}{:}\bullet\tau_{i,k_i} \vdash t_{k_i} : \tau}{\Gamma \vdash \mathit{case}\ t\ \mathit{of}\ \mathit{In}_{i,1}(x_1) \Rightarrow t_1 \mid \ldots \mid \mathit{In}_{i,k_i}(x_{k_i}) \Rightarrow t_{k_i} : \tau} \qquad \frac{\Gamma \vdash t : \tau \qquad \vdash \tau \leq \sigma}{\Gamma \vdash t : \sigma}$$

Figure 1: Typing rules of Nakano's lambda calculus

$$\frac{\vdash \tau \leq \sigma}{\vdash \bullet\tau \leq \bullet\sigma} \qquad \frac{\vdash \tau' \leq \tau \qquad \vdash \sigma \leq \sigma'}{\vdash \tau \to \sigma \leq \tau' \to \sigma'} \qquad \frac{\vdash \tau \leq \tau' \qquad \vdash \sigma \leq \sigma'}{\vdash \tau \times \sigma \leq \tau' \times \sigma'} \qquad \frac{}{\vdash \tau \leq \bullet\tau}$$

$$\frac{}{\vdash \tau \to \sigma \leq \bullet\tau \to \bullet\sigma} \qquad \frac{}{\vdash \bullet\tau \to \bullet\sigma \leq \bullet(\tau \to \sigma)} \qquad \frac{}{\vdash \tau \times \sigma \leq \bullet\tau \times \bullet\sigma} \qquad \frac{}{\vdash \bullet\tau \times \bullet\sigma \leq \bullet(\tau \times \sigma)}$$

Figure 2: Subtyping

2 Nakano's Lambda Calculus

In this section we recall the typed lambda calculus of Nakano [8]. It allows certain forms of guarded recursive definitions, and tracks guardedness via a modality in the type system. We shall show that this modality can be understood semantically as a scaling operation of the distances in a metric space.

The precise language considered below differs from that presented by Nakano in [8] in that we assume a fixed number of "global" (possibly mutually recursive) datatype declarations. In contrast, Nakano includes equi-recursive types of the form $\mu X.A$ (subject to some well-formedness conditions that ensure formal contractiveness, and a type equivalence relation).

Syntax and typing. We assume that there is a fixed number of type identifiers $ty_1, \ldots, ty_n$, each associated with a recursive data type equation in the style of Haskell or ML that specifies constructors $\mathit{In}_{i,1}, \ldots, \mathit{In}_{i,k_i}$:

$$\mathit{data}\ ty_i = \mathit{In}_{i,1}\ \mathit{of}\ \tau_{i,1} \mid \ldots \mid \mathit{In}_{i,k_i}\ \mathit{of}\ \tau_{i,k_i}\ . \tag{1}$$

The types $\tau_{i,1}, \ldots, \tau_{i,k_i}$ that appear on the right hand side of (1) are built up from ground types b (for simplicity, we restrict ourselves to Int) and the identifiers $ty_1, \ldots, ty_n$, using product and function space and the unary type constructor '$\bullet$' (referred to as 'later' modality in recent work on step-indexed semantics [3]). One concrete instance of (1) is a type of infinite sequences of τ's: $\mathit{data}\ \mathit{seq}_\tau = \mathit{Cons}\ \mathit{of}\ \tau \times \mathit{seq}_\tau$. In general, these declarations need not be monotone, however, and the identifiers ty_i may also appear in negative positions. For instance, Nakano [8, Ex. 1] considers the type $\mathit{data}\ u = \mathit{Fold}\ \mathit{of}\ u \to \tau$. With this type, the fixed point combinator Y from untyped lambda calculus has type $(\bullet\tau \to \tau) \to \tau$; thus, it is not necessary to include a primitive for recursion.

Terms and their types are given in Figure 1. The type system uses the subtyping relation defined in Figure 2 (omitting the reflexivity and transitivity rules). Intuitively, the type $\bullet\tau$ may be understood as the set of values of type τ that need to be guarded by a constructor. This intuition explains the two rules

for typing constructor applications and pattern matching in Figure 1: The application of a constructor lets one remove the outermost modal operator, and it also folds the recursive type. Conversely, the *case* construct removes a constructor, and the typing rule records this fact by applying the modality to the type of the bound variables. The typing rules for function application and component projections are also generalized to take care of the modal operator. We write $Tm(\tau)$ for the set of closed terms of type τ.

Operational semantics and contextual equivalence. The operational semantics consists of the usual (deterministic, call-by-name) reduction rules of lambda calculus with constructor types. One does not reduce under constructors: every term of the form $In_{i,j}(t)$ is a value. We take contextual equivalence as our notion of program equivalence. Formally, we observe the values that terms yield when plugged into closing, base-type valued contexts: Let us write $\vdash C : (\Gamma' \triangleright \tau') \multimap (\Gamma \triangleright \tau)$ for a context C if $\Gamma \vdash C[t] : \tau$ whenever $\Gamma' \vdash t : \tau'$. Then t_1 and t_2 are *contextually equivalent*, written $\Gamma \vdash t_1 \simeq t_2 : \tau$, if $\Gamma \vdash t_1 : \tau$ and $\Gamma \vdash t_2 : \tau$, and for all $\vdash C : (\Gamma \triangleright \tau) \multimap (\emptyset \triangleright Int)$ and $n \in \mathbb{N}$, $C[t_1] \xrightarrow{*} \underline{n} \Leftrightarrow C[t_2] \xrightarrow{*} \underline{n}$.

3 Denotational Semantics

We give a denotational semantics of types and terms in the category of complete, 1-bounded ultrametric spaces and non-expansive maps between them.

Ultrametric spaces. We briefly recall some basic definitions and results about metric spaces [10]. A *1-bounded ultrametric space* (X,d) is a metric space where the distance function $d : X \times X \to \mathbb{R}$ takes values in the closed interval $[0,1]$ and satisfies the "strong" triangle inequality $d(x,y) \leq \max\{d(x,z),d(z,y)\}$. A metric space is *complete* if every Cauchy sequence has a limit. Because of the use of max in the above inequality, rather than '+' as for the weaker triangle inequality of metric spaces in general, a sequence $(x_n)_{n\in\mathbb{N}}$ in an ultrametric space (X,d) is a Cauchy sequence if for every $\varepsilon > 0$ there exists $n \in \mathbb{N}$ such that $d(x_k,x_{k+1}) \leq \varepsilon$ for every $k \geq n$.

A function $f : X_1 \to X_2$ between metric spaces (X_1,d_1), (X_2,d_2) is *non-expansive* if $d_2(f(x),f(y)) \leq d_1(x,y)$ for all $x,y \in X_1$. It is *contractive* if there exists some $\delta < 1$ such that $d_2(f(x),f(y)) \leq \delta \cdot d_1(x,y)$ for all $x,y \in X_1$. The complete, 1-bounded, non-empty, ultrametric spaces and non-expansive functions between them form a Cartesian closed category $CBUlt_{ne}$. Products are given by the set-theoretic product where the distance is the maximum of the componentwise distances. The exponential $(X_1,d_1) \to (X_2,d_2)$ has the set of non-expansive functions from (X_1,d_1) to (X_2,d_2) as underlying set, and the distance function is given by $d_{X_1\to X_2}(f,g) = \sup\{d_2(f(x),g(x)) \mid x \in X_1\}$.

A functor $F : (CBUlt_{ne}^{\mathrm{op}})^n \times CBUlt_{ne}^n \longrightarrow CBUlt_{ne}$ is *locally non-expansive* if $d(F(f,g),F(f',g')) \leq \max\{d(f_1,f_1'),d(g_1,g_1'),\ldots,d(f_n,f_n'),d(g_n,g_n')\}$ for all non-expansive $f = (f_1,\ldots,f_n)$, $f' = (f_1',\ldots,f_n')$, $g = (g_1,\ldots,g_n)$ and $g' = (g_1',\ldots g_n')$. It is *locally contractive* if there exists some $\delta < 1$ such that $d(F(f,g),F(f',g')) \leq \delta \cdot \max\{d(f_1,f_1'),d(g_1,g_1'),\ldots,d(f_n,f_n'),d(g_n,g_n')\}$. Note that the factor δ is the same for all hom-sets. By multiplication of the distances of (X,d) with a non-negative shrinking factor $\delta < 1$, one obtains a new ultrametric space, $\delta \cdot (X,d) = (X,d')$ where $d'(x,y) = \delta \cdot d(x,y)$. By shrinking, a locally non-expansive functor F yields a locally contractive functor $(\delta \cdot F)(X,Y) = \delta \cdot (F(X,Y))$.

It is well-known that one can solve recursive domain equations in $CBUlt_{ne}$ by an adaptation of the inverse-limit method from classical domain theory:

Theorem 1 (America-Rutten [1]). *Let $F_i : (CBUlt_{ne}^{\mathrm{op}})^n \times CBUlt_{ne}^n \longrightarrow CBUlt_{ne}$ be locally contractive functors for $i = 1,\ldots,n$. Then there exists a unique (up to isomorphism) $X = (X_k,d_k)_k \in CBUlt_{ne}^n$ such that $(X_i,d_i) \cong F_i(X,X)$ for all i.*

The metric spaces we consider below are *bisected*, meaning that all non-zero distances are of the form 2^{-n} for some natural number $n \geq 0$. When working with bisected metric spaces, the notation $x \overset{n}{=} y$ means that $d(x,y) \leq 2^{-n}$. Each relation $\overset{n}{=}$ is an equivalence relation because of the ultrametric inequality; we are therefore justified in referring to the relation $\overset{n}{=}$ as "n-equality." Since the distance of a bisected metric space is bounded by 1, the relation $x \overset{0}{=} y$ always holds. Moreover, the n-equalities become finer as n increases, and $x = y$ if $x \overset{n}{=} y$ holds for all n. Finally, a function $f : X_1 \to X_2$ between bisected metric spaces is non-expansive iff $x_1 \overset{n}{=} x_1' \Rightarrow f(x_1) \overset{n}{=} f(x_1')$, and contractive iff $x_1 \overset{n}{=} x_1' \Rightarrow f(x_1) \overset{n+1}{=} f(x_1')$ for all n.

Denotational semantics of Nakano's lambda calculus. We can now define the interpretation of types and terms in $CBUlt_{ne}$. More precisely, by induction on the type τ we define locally non-expansive functors $F_\tau : (CBUlt_{ne}^{op})^n \times CBUlt_{ne}^n \longrightarrow CBUlt_{ne}$, by separating positive and negative occurrences of the identifiers $ty_1, \ldots, ty_n$ in τ:

$$\begin{aligned} F_{Int}(X,Y) &= (\mathbb{Z}, d), \text{ where } d \text{ is the discrete metric} \\ F_{ty_i}(X,Y) &= Y_i \\ F_{\tau_1 \times \tau_2}(X,Y) &= F_{\tau_1}(X,Y) \times F_{\tau_2}(X,Y) \\ F_{\tau_1 \to \tau_2}(X,Y) &= F_{\tau_1}(Y,X) \to F_{\tau_2}(X,Y) \\ F_{\bullet\tau}(X,Y) &= \tfrac{1}{2} \cdot F_\tau(X,Y) \end{aligned}$$

Now consider the functors $F_i : (CBUlt_{ne}^{op})^n \times CBUlt_{ne}^n \longrightarrow CBUlt_{ne}$ for $i = 1, \ldots, n$, defined by

$$F_i(X,Y) = \tfrac{1}{2} \cdot F_{\tau_{i,1}}(X,Y) + \ldots + \tfrac{1}{2} \cdot F_{\tau_{i,k_i}}(X,Y) \; . \tag{2}$$

Because of the shrinking factor $1/2$ in (2), each F_i is in fact locally contractive. Theorem 1 therefore gives a unique fixed point D with $\iota_i : F_i(D,D) \cong D_i$ for all i. We use D to give the semantics of types:

$$[\![\tau]\!] \overset{def}{=} F_\tau(D,D) \; .$$

Example 2 (Interpretation of streams). On streams, the semantics yields the "natural" metric. In fact, since $[\![seq_\tau]\!] \cong \frac{1}{2} \cdot ([\![\tau]\!] \times [\![seq_\tau]\!])$ we have

$$d_{[\![seq_\tau]\!]}(s_1 s_2 \ldots, s_1' s_2' \ldots) \leq 2^{-n} \;\Leftrightarrow\; d_{[\![\tau]\!]}(s_1, s_1') \leq 2^{-(n-1)} \wedge d_{[\![seq_\tau]\!]}(s_2 s_3 \ldots, s_2' s_3' \ldots) \leq 2^{-(n-1)} \; .$$

In particular, because of the discrete metric on $[\![Int]\!]$, for seq_{Int} this means $s \overset{n}{=} s'$ holds if $s_1 = s_1', \ldots, s_{n-1} = s_{n-1}'$, i.e., if the common prefix of s and s' has at least length $n-1$, and $s = s'$ if $s_i = s_i'$ for all $i \in \mathbb{N}$.

For each subtyping derivation Δ of a judgement $\vdash \tau \leq \sigma$ we define a corresponding coercion function, i.e., a non-expansive function $[\![\Delta]\!] : [\![\tau]\!] \to [\![\sigma]\!]$. First note that, for each of the 5 subtyping axioms $\vdash \tau \leq \sigma$ in Figure 2 (as well as the reflexivity axiom), the underlying sets of $[\![\tau]\!]$ and $[\![\sigma]\!]$ are equal. Thus we can define $[\![\Delta]\!]$ as the identity on the underlying sets, and it is easy to check that Δ is in fact non-expansive.[1] If Δ is obtained from Δ_1 and Δ_2 by an application of the transitivity rule, then $[\![\Delta]\!]$ is defined as the composition of $[\![\Delta_1]\!]$ and $[\![\Delta_2]\!]$. Finally, for each of the 3 structural rules in Figure 2, we use the functorial property of the respective type constructor to define $[\![\Delta]\!]$ from the coercions of the subderivations. It follows by induction that the coercion determined from any derivation Δ of $\vdash \tau \leq \sigma$ is the identity on the underlying set of $[\![\tau]\!]$, and hence independent of Δ.

[1] Note however that even though the underlying sets are the same, the inclusion of $[\![\bullet(\tau \to \sigma)]\!]$ into $[\![\bullet\tau \to \bullet\sigma]\!]$ fails to be non-expansive. Thus we cannot have $\bullet(\tau \to \sigma) \leq \bullet\tau \to \bullet\sigma$ in general.

Each term $x_1{:}\tau_1, \ldots, x_n{:}\tau_n \vdash t : \tau$ denotes a non-expansive function

$$[\![t]\!] : [\![\tau_1]\!] \times \ldots \times [\![\tau_n]\!] \to [\![\tau]\!]$$

which is defined by induction on the typing derivation. The cases of lambda abstraction, application, pairing and projections are given in terms of the cartesian closed structure on $CBUlt_{ne}$. From the definition of the type interpretation it follows that ι_i is an isomorphism between $\frac{1}{2} \cdot [\![\tau_{i,1}]\!] + \ldots + \frac{1}{2} \cdot [\![\tau_{i,k_i}]\!]$ and $[\![ty_i]\!]$. Together with the injections from $[\![\bullet\, \tau_{i,j}]\!] = \frac{1}{2} \cdot [\![\tau_{i,j}]\!]$ into the sum, this isomorphism is used to interpret the constructors and pattern matching of the calculus:

$$[\![In_{i,j}(t)]\!]\,(\vec{a}) = (\iota_i \circ \mathrm{in}_j \circ [\![t]\!])(\vec{a}) \quad \text{and} \quad [\![case\ t\ of\ In_{i,1}(x_1) \Rightarrow t_1 \mid \ldots \mid In_{i,k_i}(x_{k_i}) \Rightarrow t_{k_i}]\!]\,(\vec{a}) = [\![t_j]\!]\,(\vec{a}, a)$$

where $\iota_i^{-1}([\![t]\!]\,(\vec{a})) = \mathrm{in}_j(a)$ for some $1 \leq j \leq k_i$ and $a \in [\![\tau_{i,j}]\!]$. We obtain a model that is sound:

Theorem 3 (Soundness)**.** *If $\emptyset \vdash t : \tau$ and $t_1 \to t_2$ then $[\![t_1]\!] = [\![t_2]\!]$ in $[\![\tau]\!]$.*

Remark 4 (Recursive definitions). In [8] Nakano shows that the fixed point combinator Y from untyped lambda calculus can be represented by $\lambda f.\Delta_f\,(Fold(\Delta_f))$ where $\Delta_f = \lambda x.f(case\ x\ of\ Fold(y) \Rightarrow yx)$ and the data type $data\ u = Fold\ of\ u \to \tau$ is assumed, and that this term has type $(\bullet\,\tau \to \tau) \to \tau$. Note that in the above interpretation, where the modality is interpreted by scaling the distances by $1/2$, the type $\bullet\,\tau \to \tau$ denotes the set of all non-expansive functions $\frac{1}{2} \cdot [\![\tau]\!] \to [\![\tau]\!]$, or equivalently (by the bisectedness of $[\![\tau]\!]$) the contractive functions on $[\![\tau]\!]$. Such functions have a unique fixed point in $[\![\tau]\!]$ by the Banach fixed point theorem, and we can show that Y indeed returns this fixed point since $f([\![Y]\!]\,f) = [\![Y]\!]\,f$.

As an alternative to this coding, we could introduce a recursion operator $rec : (\bullet\,\tau \to \tau) \to \tau$ for each τ as a primitive, with reduction rule $rec\,t \to t\,(rec\,t)$, and use the above observation for its interpretation.

Computational adequacy. We now relate the interpretation of lambda terms in the metric model and their operational behaviour. In particular, we prove that the semantics is sound for reasoning about contextual equivalence: *if two terms have the same denotation then they are contextually equivalent*.

The general idea of the adequacy proof is standard: the universal quantification over contexts prevents a direct inductive proof, so we use the compositionality of the denotational semantics and construct a (Kripke) logical relation between semantics and syntax. More precisely, we consider the family $(R^k_\tau)^k_\tau$ of relations indexed by types τ and natural numbers k, where $R^k_\tau \subseteq [\![\tau]\!] \times Tm(\tau)$ is given by:

$$\begin{aligned}
n\ R^k_{Int}\ t &\Leftrightarrow t \xrightarrow{*} \underline{n} \\
(a_1, a_2)\ R^k_{\tau_1 \times \tau_2}\ t &\Leftrightarrow a_1\ R^k_{\tau_1}\ proj_1(t) \wedge a_2\ R^k_{\tau_2}\ proj_2(t) \\
f\ R^k_{\tau_1 \to \tau_2}\ t &\Leftrightarrow \forall j, a_1, t_1.\ j \leq k \wedge a_1\ R^j_{\tau_1}\ t_1 \Rightarrow f a_1\ R^j_{\tau_2}\ t\, t_1 \\
a\ R^k_{\bullet\tau}\ t &\Leftrightarrow k > 0 \Rightarrow a\ R^{k-1}_\tau\ t \\
a\ R^k_{ty_i}\ t &\Leftrightarrow \exists a', t'.\ a = (\iota_i \circ \mathrm{in}_j)(a') \wedge t \xrightarrow{*} In_{i,j}(t') \wedge a'\ R^k_{\bullet\tau_{i,j}}\ t'
\end{aligned}$$

Note that it is the natural number index which lets us define the relations R^k_τ inductively in the presence of recursive types ty_i. We prove the 'fundamental property' by induction on typing derivations:

Proposition 5. *If $\Gamma \vdash t : \tau$, $k \in \mathbb{N}$, and $a_i\ R^k_{\tau_i}\ t_i$ for all $x_i{:}\tau_i$ in Γ, then $[\![t]\!]\,(\vec{a})\ R^k_\tau\ t[\vec{x}{:=}\vec{t}]$.*

Proof sketch. By induction on the derivation of $\Gamma \vdash t : \tau$. The proof uses the closure of the relations under the operational semantics, the downwards closure (Kripke monotonicity) in k, and the closure under the coercions determined by the subtyping relation. □

Now adequacy is easily proved. In fact, given $\Gamma \vdash t_1 : \tau$, $\Gamma \vdash t_2 : \tau$ and $\vdash C : (\Gamma \triangleright \tau) \multimap (\emptyset \triangleright \mathit{Int})$, and $C[t_1] \xrightarrow{*} \underline{n}$ for some $n \in \mathbb{N}$, then $[\![C[t_2]]\!] = [\![C[t_1]]\!] = n$ follows from the assumption $[\![t_1]\!] = [\![t_2]\!]$ and Theorem 3. By Proposition 5, $[\![C[t_2]]\!]\, R^k_{\mathit{Int}}\, C[t_2]$, and therefore $n\, R^k_{\mathit{Int}}\, C[t_2]$ holds. But by definition this means $C[t_2] \xrightarrow{*} \underline{n}$, and with a symmetric argument the claim $[\![t_1]\!] = [\![t_2]\!] \;\Rightarrow\; \Gamma \vdash t_1 \simeq t_2 : \tau$ follows.

Apart from using the semantics directly to reason about contextual equivalence, we can also use computational adequacy to derive more abstract proof principles from it. For instance, by the characterization of the metric on $\mathit{seq}_{\mathit{Int}}$ given in Example 2 and the fact that there are closed terms $\mathit{get}_n : \mathit{seq}_{\mathit{Int}} \to \bullet^n \mathit{Int}$ that yield the n-th element of a sequence, we obtain a variant of Bird and Wadler's *take lemma*: if $\mathit{get}_n\, t_1$ and $\mathit{get}_n\, t_2$ reduce to the same value, for each $n \in \mathbb{N}$, then $\emptyset \vdash t_1 \simeq t_2 : \mathit{seq}_{\mathit{Int}}$. Similarly, we can exploit the uniqueness of fixed points of contractive equations also in the operational setting: if there exists a closed term $f : \bullet\tau \to \tau$ such that $f\, t_1$ and t_1 are convertible, and also $f\, t_2$ and t_2 are convertible, then $\emptyset \vdash t_1 \simeq t_2 : \tau$. See, for instance, Hinze's article [6] for numerous applications of such a unique fixed point principle phrased in the context of Haskell streams, and Pottier's work [9] for a similar application to establish type equivalences in the context of F_ω with recursive kinds.

4 Related Work

Metric semantics of PCF. Escardó [5] presents a metric model of PCF. One can, almost,[2] factor his interpretation of PCF into two parts: (1) a syntactic translation from PCF to Nakano's calculus, extended with constants for integer operations and a booleans, and (2) the metric interpretation of Nakano's calculus presented here. The basic idea of the syntactic translation is that a potentially divergent PCF term of integer type is translated into a term of the recursive type $\mathit{Int}_s = \mathit{Done\ of\ Int} \mid \mathit{Next\ of\ Int}_s$. After evaluating such a term, one either obtains an actual integer answer, *Done(n)*, or a new term that one can evaluate in the hope of eventually getting an integer answer. PCF function types are translated to function types.

Now, for all types τ that are used to interpret PCF types, one defines a term $\mathit{delay} : \bullet\tau \to \tau$; this term is used to define the fixed-point combinator of PCF in terms of the fixed-point combinator of our calculus (Remark 4). See Escardó for more details. Intuitively, the idea is that unfolding a recursive definition takes a "computation step," which will be visible as an extra *Next* in the final answer of type Int_s. Escardó shows an adequacy result which implies that the semantics of a PCF term does indeed match the number of unfoldings of the fixed-point combinator; the same information can be obtained from our adequacy proof using the definition of the logical relation on recursive types. Escardó gives two adequacy proofs, the first of which uses a Kripke logical relation as in our adequacy proof.

Since Escardó considers PCF, he does not treat recursively defined types, as we do here.

5 Conclusion and Future Work

We have presented a computationally adequate metric model of lambda calculus with guarded recursion. It complements Nakano's realizability interpretations [8], and it explains the modality in terms of scaling.

We conjecture that an adaptation of the present model can be used to give a model for focusing proof systems with recursive types [11].

[2] Due to the syntactic restrictions on recursive types in our variant of Nakano's calculus, the metrics on PCF ground types (and hence on all types) differ slightly in our model and Escardó's model.

References

[1] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *Journal of Computer and System Sciences*, 39(3):343–375, 1989.

[2] A. W. Appel and D. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Transactions on Programming Languages and Systems*, 23(5):657–683, 2001.

[3] A. W. Appel, P.-A. Melliès, C. D. Richards, and J. Vouillon. A very modal model of a modern, major, general type system. In *Proceedings of POPL*, pages 109–122, 2007.

[4] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *Proceedings of FOSSACS*, pages 456–470, 2009.

[5] M. Escardó. A metric model of PCF. Presented at the Workshop on Realizability Semantics and Applications, June 1999. Available at the author's web page.

[6] R. Hinze. Functional pearl: streams and unique fixed points. In *Proceedings of ICFP*, pages 189–200, 2008.

[7] N. Krishnaswami, L. Birkedal, and J. Aldrich. Verifying event-driven programs using ramified frame properties. In *Proceedings of TLDI*, pages 63–76, 2010.

[8] H. Nakano. A modality for recursion. In *Proceedings of LICS*, pages 255–266, 2000.

[9] F. Pottier. A typed store-passing translation for general references. Submitted, Apr. 2010.

[10] M. B. Smyth. Topology. In *Handbook of Logic in Computer Science*, volume 1. Oxford Univ. Press, 1992.

[11] N. Zeilberger. *The Logical Basis of Evaluation Order and Pattern-Matching*. PhD thesis, Carnegie Mellon University, 2009.

A Step-indexed Kripke Model of Hidden State via Recursive Properties on Recursively Defined Metric Spaces

Lars Birkedal
IT University of Copenhagen
birkedal@itu.dk

Jan Schwinghammer
Saarland University
jan@ps.uni-saarland.de

Kristian Støvring
IT University of Copenhagen
kss@itu.dk

Abstract

Frame and anti-frame rules have been proposed as proof rules for modular reasoning about programs. Frame rules allow one to hide irrelevant parts of the state during verification, whereas the anti-frame rule allows one to hide local state from the context. We give a possible worlds semantics for Charguéraud and Pottier's type and capability system including frame and anti-frame rules, based on the operational semantics and step-indexed heap relations. The worlds are constructed as a recursively defined predicate on a recursively defined metric space, which provides a considerably simpler alternative compared to a previous construction.

1 Introduction

Reasoning about higher-order stateful programs is notoriously difficult, and often involves the need to track aliasing information. A particular line of work that addresses this point are substructural type systems with regions, capabilities and singleton types [1, 7, 8]. In this context, Pottier [9] presented the anti-frame rule as a proof rule for hiding local state. It states that (the description of) a piece of mutable state that is *local* to a procedure can be removed from the procedure's external interface (expressed in the type system). The benefits of hiding local state include simpler interface specifications, simpler reasoning about client code, and fewer restrictions on the procedure's use because of potential aliasing. Thus, in combination with frame rules that allow the irrelevant parts of the state to be hidden during verification, the anti-frame rule provides an important ingredient for modular reasoning about programs.

Soundness of the frame and anti-frame rules is subtle, and relies on properties of the programming language. Pottier [9] sketched a proof for the anti-frame rule by a progress and preservation argument, resting on assumptions about the existence of certain recursively defined types. Subsequently, Schwinghammer et al. [12] investigated the semantic foundations of the anti-frame rule by identifying sufficient conditions for its soundness, and by instantiating their general setup to prove soundness for a separation logic variant of the rule. The latter was done by giving a Kripke model where assertions are indexed over recursively defined worlds. The recursive domain equation involved functions that should be monotonic with respect to an order relation that is specified using the isomorphism of the solution itself and an operator on the recursively defined worlds. This means that the standard existence theorems do not appear to apply, and thence we had to give the solution by a laborious explicit inverse-limit construction [12].

Here we explore an alternative approach, which consists of two steps. First, we consider a recursive metric space domain equation without any monotonicity requirement, for which we obtain a solution by appealing to a standard existence theorem. Second, we carve out a suitable subset of what might be called *hereditarily monotonic* functions. We show how to define this recursively specified subset. The resulting subset of monotonic functions is, however, not a solution to the original recursive domain equation; hence we verify that the semantic constructions used to justify the anti-frame rule in [12] suitably restrict to the recursively defined subset of hereditarily monotonic functions. In summa, this results in a considerably simpler model construction than the earlier one in *loc. cit.*

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 27-33

In the next section we give a brief overview of Charguéraud and Pottier's type and capability system [7, 9] with higher-order frame and anti-frame rules. Section 3 summarizes some background on ultrametric spaces and presents the construction of a set of hereditarily monotonic recursive worlds. Following recent work by Birkedal et al. [4], we work with step-indexed heap relations based on the operational semantics of the calculus. The worlds thus constructed are then used (Section 4) to give a model of the type and capability system. For space reasons, many details are relegated to an online technical appendix.

2 A Calculus of Capabilities

Syntax and operational semantics. We consider a standard call-by-value, higher-order language with general references, sum and product types, and polymorphic and recursive types. For concreteness, the following grammar gives the syntax of values and expressions, keeping close to the notation of [7, 9]:

$$v ::= x \mid () \mid \mathsf{inj}^i\, v \mid (v_1, v_2) \mid \mathsf{fun}\, f(x){=}t \mid l \qquad t ::= v \mid (v\,t) \mid \mathsf{case}(v_1, v_2, v) \mid \mathsf{proj}^i\, v \mid \mathsf{ref}\, v \mid \mathsf{get}\, v \mid \mathsf{set}\, v$$

Here, the term $\mathsf{fun}\, f(x){=}t$ stands for the recursive procedure f with body t. The operational semantics is given by a relation $(t \mid h) \longmapsto (t' \mid h')$ between configurations that consist of a (closed) expression t and a heap h. We take a heap h to be a finite map from locations l to closed values, we use the notation $h\#h'$ to indicate that two heaps h, h' have disjoint domains, and we write $h \cdot h'$ for the union of two such heaps. By *Val* we denote the set of closed values.

Types. Charguéraud and Pottier's type system uses *capabilities*, *value types*, and *memory types*. A capability C describes a heap property, much like the assertions of a Hoare-style program logic. For instance, $\{\sigma : \mathsf{ref}\,\mathsf{int}\}$ asserts that σ is a valid location that contains an integer value. More complex assertions can be built by separating conjunctions $C_1 * C_2$ and universal and existential quantification over names σ. Value types τ classify values; they include base types, singleton types $[\sigma]$, and are closed under products, sums, and universal quantification. *Memory types* χ, θ describe the result of computations. They extend the value types by a type of references, and also include all types of the form $\exists\vec{\sigma}.\tau * C$ which describe the final value and heap that result from the evaluation of an expression. Arrow types (which are value types) have the form $\chi_1 \to \chi_2$ and thus, like the pre- and post-conditions of a triple in Hoare logic, make explicit which part of the heap is accessed and modified when a procedure is called. We also allow recursive capabilities, value types, and memory types, resp., provided the recursive definition is (formally) contractive, i.e., the recursion must go through a type constructor such as $\to$.

Frame and anti-frame rules. Each of the syntactic categories is equipped with an *invariant extension* operation, $\cdot \otimes C$. Intuitively, this operation conjoins C to the domain and codomain of every arrow type that occurs within its left hand argument, which means that the capability C is preserved by all procedures of this type. This intuition is made precise by regarding capabilities and types modulo a structural equivalence which subsumes the "distribution axioms" for $\otimes$ that are used to express generic higher-order frame rules [6]. Two key cases of the structural equivalence are the distribution axioms for arrow types, $(\chi_1 \to \chi_2) \otimes C = (\chi_1 \otimes C * C) \to (\chi_2 \otimes C * C)$, and for successive extensions, $(\chi \otimes C_1) \otimes C_2 = \chi \otimes (C_1 \circ C_2)$ where the derived operation $C_1 \circ C_2$ abbreviates the conjunction $(C_1 \otimes C_2) * C_2$.

There are two typing judgements, $x_1{:}\tau_1, \ldots, x_n{:}\tau_n \vdash v : \tau$ for values, and $x_1{:}\chi_1, \ldots, x_n{:}\chi_n \Vdash t : \chi$ for expressions. The latter is similar to a Hoare triple where (the separating conjunction of) $\chi_1, \ldots, \chi_n$ serves as a precondition and χ as a postcondition. This view provides some intuition for the following "shallow"

and "deep" frame rules, and for the (roughly dual) anti-frame rule:

$$[SF]\frac{\Gamma \Vdash t : \chi}{\Gamma * C \Vdash t : \chi * C} \qquad [DF]\frac{\Gamma \Vdash t : \chi}{(\Gamma \otimes C) * C \Vdash t : (\chi \otimes C) * C} \qquad [AF]\frac{\Gamma \otimes C \Vdash t : (\chi \otimes C) * C}{\Gamma \Vdash t : \chi} \quad (1)$$

As in separation logic, the frame rules can be used to add a capability C (which might assert the existence of an integer reference, say) as an invariant to a specification $\Gamma \Vdash t : \chi$, which is useful for local reasoning. The difference between the shallow variant $[SF]$ and the deep variant $[DF]$ is that the former adds C only on the top-level, whereas the latter also extends all arrow types nested inside Γ and χ, via $\cdot \otimes C$. While the frame rules can be used to reason about certain forms of information hiding [6], the anti-frame rule expresses a hiding principle more directly: the capability C can be removed from the specification if C is an invariant that is established by t, expressed by $\cdot * C$, and guaranteed to hold whenever control passes from t to the context and back, expressed by $\cdot \otimes C$.

3 Hereditarily Monotonic Recursive Worlds

Intuitively, capabilities describe heaps. A key idea of the model that we present next is that capabilities (as well as types and type contexts) are parameterized by invariants – this will make it easy to interpret the invariant extension operation $\otimes$, as in [11, 12]. But, as the frame and anti-frame rules in (1) indicate, invariants can be arbitrary capabilities again. Thus, we are led to consider a Kripke model where the worlds are *recursively defined*: to a first approximation, we need a solution to the equation

$$W \;=\; W \to \mathit{Pred}(\mathit{Heap})\;. \quad (2)$$

In fact, we will also need to consider a preorder on W and ensure that the interpretation of capabilities and types is *monotonic*. We will find a solution to a suitable variant of (2) using ultrametric spaces.

Ultrametric spaces. We recall some basic definitions and results about ultrametric spaces; for a less condensed introduction to ultrametric spaces we refer to [13]. A *1-bounded ultrametric space* (X,d) is a metric space where the distance function $d : X \times X \to \mathbb{R}$ takes values in the closed interval $[0,1]$ and satisfies the "strong" triangle inequality $d(x,y) \leq \max\{d(x,z),d(z,y)\}$. A metric space is *complete* if every Cauchy sequence has a limit. A function $f : X_1 \to X_2$ between metric spaces (X_1,d_1), (X_2,d_2) is *non-expansive* if $d_2(f(x),f(y)) \leq d_1(x,y)$ for all $x,y \in X_1$. It is *contractive* if there exists some $\delta < 1$ such that $d_2(f(x),f(y)) \leq \delta \cdot d_1(x,y)$ for all $x,y \in X_1$. By multiplication of the distances of (X,d) with a non-negative factor $\delta < 1$, one obtains a new ultrametric space, $\delta \cdot (X,d) = (X,d')$ where $d'(x,y) = \delta \cdot d(x,y)$.

The complete, 1-bounded, non-empty, ultrametric spaces and non-expansive functions between them form a Cartesian closed category CBUlt_{ne}. Products are given by the set-theoretic product where the distance is the maximum of the componentwise distances. The exponential $(X_1,d_1) \to (X_2,d_2)$ has the set of non-expansive functions from (X_1,d_1) to (X_2,d_2) as underlying set, and the distance function is given by $d_{X_1 \to X_2}(f,g) = \sup\{d_2(f(x),g(x)) \mid x \in X_1\}$.

The notation $x \overset{n}{=} y$ means that $d(x,y) \leq 2^{-n}$. Each relation $\overset{n}{=}$ is an equivalence relation because of the ultrametric inequality; we refer to the relation $\overset{n}{=}$ as "n-equality." Since the distances are bounded by 1, $x \overset{0}{=} y$ always holds, and the n-equalities become finer as n increases. If $x \overset{n}{=} y$ holds for all n then $x = y$.

Uniform predicates, worlds and world extension. Let $(A,\sqsubseteq)$ be a partially ordered set. An *upwards closed, uniform predicate* on A is a subset $p \subseteq \mathbb{N} \times A$ that is downwards closed in the first and upwards closed in the second component: if $(k,a) \in p$, $j \leq k$ and $a \sqsubseteq b$, then $(j,b) \in p$. We write $\mathit{UPred}(A)$ for the set of all upwards closed, uniform predicates on A, and we define $p_{[k]} = \{(j,a) \mid j < k\}$. Note that

$p_{[k]} \in UPred(A)$. We equip $UPred(A)$ with the distance function $d(p,q) = \inf\{2^{-n} \mid p_{[n]} = q_{[n]}\}$, which makes $(UPred(A), d)$ an object of $CBUlt_{ne}$.

In our model, we use $UPred(A)$ with the following concrete instances for the partial order $(A, \sqsubseteq)$: (1) *heaps* $(Heap, \sqsubseteq)$, where $h \sqsubseteq h'$ iff $h' = h \cdot h_0$ for some $h_0 \# h$, (2) *values* $(Val, \sqsubseteq)$, where $u \sqsubseteq v$ iff $u = v$, and (3) *stateful values* $(Val \times Heap, \sqsubseteq)$, where $(u,h) \sqsubseteq (v,h')$ iff $u = v$ and $h \sqsubseteq h'$. We also use variants of the latter two instances where the set *Val* is replaced by the set of value substitutions, *Env*, and by the set of closed expressions, *Exp*. On $UPred(Heap)$, ordered by subset inclusion, we have a complete Heyting BI algebra structure [3]. Below we only need the separating conjunction and its unit I, given by

$$p_1 * p_2 \;=\; \{(k,h) \mid \exists h_1, h_2.\; h = h_1 \cdot h_2 \;\wedge\; (k,h_1) \in p_1 \wedge\; (k,h_2) \in p_2\} \quad \text{and} \quad I \;=\; \mathbb{N} \times Heap\;.$$

It is well-known that one can solve recursive domain equations in $CBUlt_{ne}$ by an adaptation of the inverse-limit method from classical domain theory [2]. In particular, with regard to (2) above:

Theorem 1. *There exists a unique (up to isomorphism)* $(X,d) \in CBUlt_{ne}$ *s.t.* $\iota : \frac{1}{2} \cdot X \to UPred(Heap) \cong X$.

Using the pointwise lifting of separating conjunction to $1/2 \cdot X \to UPred(Heap)$ we define a *composition operation* on X, which reflects the syntactic abbreviation $C_1 \circ C_2 = C_1 \otimes C_2 * C_2$ of conjoining C_1 and C_2 and additionally applying an invariant extension to C_1. Formally, $\circ : X \times X \to X$ is a non-expansive operation that for all $p,q,x \in X$ satisfies

$$\iota^{-1}(p \circ q)(x) \;=\; \iota^{-1}(p)(q \circ x) * \iota^{-1}(q)(x)\;,$$

and which can be defined by an easy application of Banach's fixed point theorem as in [11]. One can show that this operation is associative and has a left and right unit given by $emp = \iota(\lambda w.I)$; thus $(X, \circ, emp)$ is a monoid in $CBUlt_{ne}$. Using $\circ$ we define an *extension operation* $\otimes : Y^{(1/2 \cdot X)} \times X \to Y^{(1/2 \cdot X)}$ for any $Y \in CBUlt_{ne}$ by $(f \otimes x)(x') = f(x \circ x')$. Without going into details, let us remark that this operation is the semantic counterpart to the syntactic invariant extension, and thus plays a key role in explaining the frame and anti-frame rules. However, for Pottier's anti-frame rule we also need to ensure that specifications are not invalidated by invariant extension. This requirement is stated via monotonicity, as we discuss next.

Relations on ultrametric spaces and hereditarily monotonic worlds. As a conseqence of the fact that $\circ$ defines a monoid structure on X there is an induced preorder on X:

$$x \sqsubseteq y \;\Leftrightarrow\; \exists x_0.\; y = x \circ x_0\;.$$

For modelling the anti-frame rule, we aim for a set of worlds similar to $X \cong 1/2 \cdot X \to UPred(Heap)$ but where the function space consists of the non-expansive functions that are additionally monotonic, with respect to the order induced by $\circ$ on X and with respect to set inclusion on $UPred(Heap)$:

$$(W, \sqsubseteq) \;\cong\; \tfrac{1}{2} \cdot (W, \sqsubseteq) \to_{mon} (UAdm, \subseteq)\;. \tag{3}$$

Because the definition of the order $\sqsubseteq$ (induced by $\circ$) already uses the isomorphism between left-hand and right-hand side, and because the right-hand side depends on the order for the monotonic function space, the standard existence theorems for solutions of recursive domain equations do not appear to apply to (3). Previously we have constructed a solution to this equation explicitly as inverse limit of a suitable chain of approximations [12]. We show in the following that we can alternatively carve out from X a suitable subset of what we call *hereditarily monotonic* functions. This subset needs to be defined recursively.

Let $\mathscr{R}$ be the collection of all non-empty and closed relations $R \subseteq X$. Given $R \in \mathscr{R}$, we define

$$R_{[n]} \;\stackrel{def}{=}\; \{y \mid \exists x \in X.\; x \stackrel{n}{=} y \;\wedge\; x \in R\}\;.$$

Thus, $R_{[n]}$ is the set of all points within distance 2^{-n} of R. Note that $R_{[n]} \in \mathscr{R}$. In fact, $R \subseteq R_{[n]}$ by the reflexivity of n-equivalence, and if $(y_k)_{k\in\mathbb{N}}$ is a sequence in $R_{[n]}$ with limit y then $d(y_k, y) \leq 2^{-n}$ for some k, i.e., $y_k \overset{n}{=} y$. So there exists $x \in X$ with $x \in R$ and $x \overset{n}{=} y_k$, and hence $x \overset{n}{=} y$ which gives $\lim_n y_n \in R_{[n]}$.

We make some further observations that follow from the properties of n-equality on X. First, $R \subseteq S$ implies $R_{[n]} \subseteq S_{[n]}$ for any $R, S \in \mathscr{R}$. Moreover, using the fact that the n-equalities become increasingly finer it follows that $(R_{[m]})_{[n]} = R_{[\min(m,n)]}$ for all $m, n \in \mathbb{N}$, so in particular each $(\cdot)_{[n]}$ is a closure operation on $\mathscr{R}$. As a consequence, we have $R \subseteq \ldots \subseteq R_{[n]} \subseteq \ldots \subseteq R_{[1]} \subseteq R_{[0]}$. By the 1-boundedness of X, $R_{[0]} = X$ for all $R \in \mathscr{R}$. Finally, $R = S$ if and only if $R_{[n]} = S_{[n]}$ for all $n \in \mathbb{N}$.

Proposition 2. *Let $d : \mathscr{R} \times \mathscr{R} \to \mathbb{R}$ be defined by $d(R,S) = \inf\{2^{-n} \mid R_{[n]} = S_{[n]}\}$. Then $(\mathscr{R}, d)$ is a complete, 1-bounded, non-empty ultrametric space. The limit of a Cauchy chain $(R_n)_{n\in\mathbb{N}}$ with $d(R_n, R_{n+1}) \leq 2^{-n}$ is given by $\bigcap_n (R_n)_{[n]}$, and in particular $R = \bigcap_n R_{[n]}$ for any $R \in \mathscr{R}$.*

We will now define the set of hereditarily monotonic functions W as a recursive predicate on X. Let the function Φ on subsets of X be given by $\Phi(R) = \{\iota(p) \mid \forall x, x_0 \in R.\ p(x) \subseteq p(x \circ x_0)\}$. If $R \in \mathscr{R}$ then $\Phi(R)$ is non-empty and closed (i.e., Φ restricts to a function on $\mathscr{R}$), and Φ is contractive. By Proposition 2 and the Banach theorem we can now define W as the (uniquely determined) fixed point of Φ and obtain

$$w \in W \;\Leftrightarrow\; \exists p.\ w = \iota(p) \;\wedge\; \forall w, w_0 \in W.\ p(w) \subseteq p(w \circ w_0)\ .$$

Note that W thus constructed does not quite satisfy (3). We do not have an isomorphism between W and the non-expansive and monotonic functions from W (viewed as an ultrametric space itself), but rather between W and all functions from X that *restrict* to monotonic functions whenever applied to hereditarily monotonic arguments. Keeping this in mind, we abuse notation and write

$$\tfrac{1}{2}\cdot W \to_{mon} UPred(A) \overset{def}{=} \{p : \tfrac{1}{2}\cdot X \to UPred(A) \mid \forall w_1, w_2 \in W.\ p(w_1) \subseteq p(w_1 \circ w_2)\}\ .$$

Then, for our particular application of interest, we also have to ensure that all the operations restrict appropriately. First, by induction we show that for all $n \in \mathbb{N}$, if $w_1, w_2 \in W$ then $w_1 \circ w_2 \in W_{[n]}$, and this entails that the composition operation $\circ$ restricts to W. In turn, this means that the $\otimes$ operator restricts accordingly: if $w \in W$ and p is in $\frac{1}{2}\cdot W \to_{mon} UPred(A)$ then so is $p \otimes w$.

4 Possible World Semantics of Capabilities

We define semantic domains for the capabilities and types of the calculus described in Section 2,

$$\begin{aligned} Cap &= \tfrac{1}{2}\cdot W \to_{mon} UPred(Heap) \\ VT &= \tfrac{1}{2}\cdot W \to_{mon} UPred(Val) \\ MT &= \tfrac{1}{2}\cdot W \to_{mon} UPred(Val \times Heap)\ , \end{aligned}$$

so that $p \in Cap$ if and only if $\iota(p) \in W$. Next, we define operations on the semantic domains that correspond to the syntactic type and capability constructors. The most interesting of these is the one for arrow types. Given $p, q \in 1/2 \cdot X \to UPred(Val \times Heap)$, $p \to q$ in $\frac{1}{2}\cdot X \to UPred(Val)$ is defined by

$$\begin{aligned} (p \to q)(x) \overset{def}{=} \{(k, \mathsf{fun}\, f(y){=}t) \mid \forall j < k.\ \forall w \in W.\ \forall r \in UPred(Heap).\ \\ \forall (j,(v,h)) \in p(x \circ w) * \iota^{-1}(x \circ w)(emp) * r. \\ (j, (t[f{:=}\mathsf{fun}\, f(y){=}t, y{:=}v], h)) \in \mathscr{E}(q)(x \circ w) * r\}\ , \end{aligned} \tag{4}$$

where $\mathscr{E}(q)$ is the extension of a world-indexed, uniform predicate on $Val \times Heap$ to one on $Exp \times Heap$. It is here where the index is linked to the operational semantics: $(k,(t,h)) \in \mathscr{E}(q)(x)$ iff for all $j \leq k, t', h'$,

$$(t\,|\,h) \longmapsto^{j} (t'\,|\,h') \;\wedge\; (t'\,|\,h') \text{ irreducible} \;\Rightarrow\; (k-j,(t',h')) \in \bigcup_{w' \in W} q(x \circ w') * \iota^{-1}(x \circ w')(emp) \;.$$

Definition (4) realizes the key ideas of our model as follows. First, the universal quantification over $w \in W$ and subsequent use of the world $x \circ w$ builds in monotonicity, and intuitively means that $p \to q$ is parametric in (and hence preserves) invariants that have been added by the procedure's context. In particular, (4) states that procedure application preserves this invariant, when viewed as the predicate $\iota^{-1}(x \circ w)(emp)$. By also conjoining r as an invariant we "bake in" the first-order frame property, which results in a subtyping axiom $\chi_1 \to \chi_2 \leq \chi_1 * C \to \chi_2 * C$ in the type system. The existential quantification over w', in the definition of $\mathscr{E}$, allows us to absorb a part of the local heap description into the world. Finally, the quantification over indices $j < k$ in (4) achieves that $(p \to q)(x)$ is uniform. There are two explanations why we require that j be *strictly* less than k. Technically, the use of $\iota^{-1}(x \circ w)$ in the definition "undoes" the scaling by $1/2$, and $j < k$ is needed to ensure the non-expansiveness of $p \to q$ as a function $1/2 \cdot X \to UPred(Val)$. Moreover, it lets us prove the typing rule for *recursive* functions by induction on k. Intuitively, the use of $j < k$ for the arguments suffices since application consumes a step.

All these constructors restrict to Cap, VT and MT, respectively. With their help it becomes straightforward to define the interpretation of capabilities and types, and to verify that the type equivalences hold with respect to this interpretation. The semantics of typing judgements is defined in analogy to (4), but also universally quantifying over worlds and indices, and it validates the typing rules of the calculus.

5 Conclusion and Future Work

To justify proof rules that take advantage of hidden state, like the frame and anti-frame rules, one needs semantic models that adequately capture this aspect of programming languages. In this paper, we have described the construction of a suitable possible worlds model where the worlds are given by a recursively defined predicate on a recursively defined metric space. In contrast to a similar model [12] which involved a tedious explicit inverse-limit construction, the present approach uses standard existence and fixed point theorems. We believe that this method provides a realistic approach to study frame and anti-frame rules in the presence of other programming language features, and to investigate the soundness of generalizations of these rules that have recently proposed by Pottier [10].

References

[1] A. Ahmed, M. Fluet, and G. Morrisett. L3: A linear language with locations. *Fundam. Inf.*, 77(4):397–449, 2007.

[2] P. America and J. J. M. M. Rutten. Solving reflexive domain equations in a category of complete metric spaces. *J. Comput. Syst. Sci.*, 39(3):343–375, 1989.

[3] B. Biering, L. Birkedal, and N. Torp-Smith. Bi-hyperdoctrines, higher-order separation logic, and abstraction. *ACM Trans. Program. Lang. Syst.*, 29(5), 2007.

[4] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. Draft, April 2010.

[5] L. Birkedal, K. Støvring, and J. Thamsborg. Realizability semantics of parametric polymorphism, general references, and recursive types. In *FOSSACS*, pages 456–470, 2009.

[6] L. Birkedal, N. Torp-Smith, and H. Yang. Semantics of separation-logic typing and higher-order frame rules for Algol-like languages. *LMCS*, 2(5:1), 2006.

[7] A. Charguéraud and F. Pottier. Functional translation of a calculus of capabilities. In *ICFP*, pages 213–224, 2008.

[8] K. Crary, D. Walker, and G. Morrisett. Typed memory management in a calculus of capabilities. In *POPL*, pages 262–275, 1999.

[9] F. Pottier. Hiding local state in direct style: a higher-order anti-frame rule. In *LICS*, pages 331–340, 2008.

[10] F. Pottier. Generalizing the higher-order frame and anti-frame rules. Unpublished, July 2009.

[11] J. Schwinghammer, L. Birkedal, B. Reus, and H. Yang. Nested Hoare triples and frame rules for higher-order store. In *CSL*, pages 440–454, 2009.

[12] J. Schwinghammer, H. Yang, L. Birkedal, F. Pottier, and B. Reus. A semantic foundation for hidden state. In *FOSSACS*, pages 2–16, 2010.

[13] M. B. Smyth. Topology. In *Handbook of Logic in Computer Science*, volume 1. Oxford Univ. Press, 1992.

How fast can the fixpoints in modal μ calculus be reached?

Marek Czarnecki
Department of Mathematics, Computer Science and Mechanics
University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
marek.czarnecki@gmail.com

Abstract

We investigate how fast modal μ formulae may reach their fixpoints. We show a way how to construct for each ordinal number α less than ω^2, a formula which reaches its fixpoint in α steps. Our approach is based on *fuses* i.e. finite one–way counters, which allow us to control the number of uses of $\Box$ while iterating our formulae.

1 Basic notions

Researchers of the modal μ calculus were widely investigating classes of formulae which reach their fixpoints relatively fast, that is in a finite number of steps (*bounded* formulae) or in no more than ω steps (*constructive* formulae). Some interesting remarks about those classes of formulae can be found in Gaëlle Fontaine's paper [1]. In this paper we analize how fast, in general, may modal μ formulae reach their least fixpoints.

Along this paper we work with the following definition.

Definition 1. *We say that a modal μ formula reaches its fixpoint in α steps in a variable x if and only if α is the least ordinal such that for all models $\mathscr{M}$ and all valuations τ, $\varphi^{\alpha+1}_{\mathscr{M},\tau}(\varnothing) = \varphi^{\alpha}_{\mathscr{M},\tau}(\varnothing)$ holds. We will denote this fact by $\mathscr{O}_x(\varphi) = \alpha$.*

Our investigation is motivated by a question asked by Damian Niwiński whether there exists a formula which reaches its fixpoint in $\omega+1$ steps and, in a broader sense, whether it is possible to control the number of interations of formulae above ω. As a response for this question Mikołaj Bojańczyk conjectured that the formula $(\Diamond x \wedge \Box p_1 \wedge p_1) \vee (\Box x \wedge \Box p_1 \wedge \neg p_1) \vee \Box\bot$ reaches its fixpoint in exactly $\omega+1$ steps. We prove this conjecture is true and make an observation on the sense of appearances of propositional variable p_1 in this formula. This consideration lets us to think of p_1 as a *fuse*, which melts while $\Box$ is used, and which prohibits adding new points afterwards. Adding more *fuses* allows us to pass through more limit ordinals. Those propositional variables may be considered also as a finite one–way counter which, however, increases its size linearly[1] – in the sense of length of the formula – while increasing its capacity. Therefore our approach can not be extended to a construction of formulae fixing at ω^2 or further.

Now we recall some basic notions. The syntax of modal μ calculus is an extension of modal logic with the construction $\mu x.\varphi$ bounding an individual variable x, that can be applied to formulae in which every occurrence of x is in range of a positive number of negations. Semantics for this construction is the following: $[\![\mu x.\varphi]\!]_{\mathscr{M},\tau} = \bigcap\{A \subseteq |\mathscr{M}| : [\![\varphi]\!]_{\mathscr{M},\tau[x:=A]} \subseteq A\}$. One can observe that this, indeed, is the least fixpoint of the map $\varphi : \mathscr{P}(|\mathscr{M}|) \to \mathscr{P}(|\mathscr{M}|)$ such that $\varphi(A) = [\![\varphi]\!]_{\mathscr{M},\tau[x:=A]}$. We will omit subscripts, if it is clear from the context, which model and valuation we work with.

It is easy to see that to some formulae φ we can not assign any ordinal number α such that $\mathscr{O}_x(\varphi) = \alpha$. It means that there are arbitrary large ordinal numbers α and models in which those formulae reach their

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 35-39

[1] This can be improved to a logarithmic growth, but with a drawback on clearness of the reasoning, thus we will work with the linear *fuses*.

fixpoints in more than α steps. Certainly the easiest example of such a formula is $\Box x$. There is a folklore way of assigning the tree to an ordinal number. To 0 we assign just one point – the root of the tree. For a successor $\alpha+1$ we add to the preexisting tree for α one more point, which is to be a new root, and attach to it simply the previous one. For a limit ordinal λ we also take a new point, which is to be a new root, but we attach to it all the roots of the trees constructed before i.e. for $\alpha<\lambda$. A straightforward transfinite induction argument leads us to conclusion that the formula $\Box x$ reaches its fixpoint in a model assigned to λ in exactly $\lambda+1$ steps, for every limit ordinal λ. Therefore there is no ordinal that bounds the number of iterations of $\Box x$ in every model – we denote it by $\mathscr{O}_x(\Box x)=\infty$.

Before proceeding to the infinite case we introduce easy examples of formulae reaching their fixpoints in finite numbers of steps.

For $n\in\omega$ let us consider formulae $\varphi_n=\Box x\wedge\Box^{n+1}\bot$. Note that for $k\leq l$ there is $\Box^k\bot\wedge\Box^l\bot\equiv\Box^k(\Box^{l-k}\bot\wedge\bot)\equiv\Box^k\bot$ and $\Box^k\bot\vee\Box^l\bot\equiv\Box^l\bot$. Therefore, for each $n\in\omega$, every model and valuation $\varphi_n^{n+1}(\varnothing)=[\![\bigvee_{i=0}^{n+1}(\Box^{i+1}\bot\wedge\Box^{n+1}\bot)]\!]=[\![\Box^{n+1}\bot\wedge\bigvee_{i=0}^{n+1}\Box^{i+1}\bot]\!]=[\![\Box^{n+1}\bot\wedge\Box^{n+2}\bot]\!]=[\![\Box^{n+1}\bot]\!]=\varphi_n^n(\varnothing)$. This gives us the upper bound for the number of iterations needed by φ_n to reach its fixpoint. To prove that it is also the lower bound, consider the trees assigned to finite ordinals from previous paragraph.

2 *Fuses* – controlling the number of interations

Let us introduce sets of *fuses*: for $n>0$ and $0\leq i\leq n$ let $C_i^n=\neg p_1\wedge\cdots\wedge\neg p_i\wedge p_{i+1}\wedge\cdots\wedge p_n$. One can treat C_i^n as different colors as it is obvious that for every $n>0$ if $i\neq j$, then $C_i^n\wedge C_j^n\equiv\bot$. We now proceed to the general construction[2] of formulae reaching their fixpoints in α steps for $\omega\leq\alpha<\omega^2$.

$$\psi_{\omega\cdot n}=\bigvee_{i=0}^{n-1}(\Diamond x\wedge C_i^n\wedge\Box C_i^n)\vee\bigvee_{i=0}^{n-2}(\Box x\wedge C_{i+1}^n\wedge\Box C_i^n),$$

$$\psi_{\omega\cdot n+m}=\psi_{\omega\cdot n}\vee\bigvee_{i=0}^{m-1}(\Box x\wedge\bigwedge_{j=0}^{i}\Box^j C_n^n\wedge\Box^{i+1}C_{n-1}^n),$$

$$\varphi_{\omega\cdot n+m}=\psi_{\omega\cdot n+m}\vee\Box\bot\,.$$

Note that $\varphi_{\omega+1}$ is exactly the formula which, according to Mikołaj Bojańczyk's conjecture, reaches its fixpoint in $\omega+1$ steps. We now show the key lemma of the paper.

Lemma 1. *Let $k>0$, α, β such that $\omega\cdot k\leq\alpha,\beta<\omega^2$, let $\mathscr{M}=(M,R,V)$ be a model and let τ be a valuation. Then for every $a\in M$, if $a\models p_k$ and $a\in\varphi_\alpha^\beta(\varnothing)$, then $a\in\varphi_\alpha^{\omega\cdot k}(\varnothing)$.*

Proof. Fix $\mathscr{M}=(M,R,V)$, τ and $a\in M$. We proceed by induction on β.

The base step for $\beta=\omega\cdot k$ is trivial as we have:
$\forall k>0\forall\omega\cdot k\leq\alpha<\omega^2\,[(a\models p_k\wedge a\in\varphi_\alpha^{\omega\cdot k}(\varnothing))\Rightarrow a\in\varphi_\alpha^{\omega\cdot k}(\varnothing)]$.

The limit step is obvious as well:
Let λ be a limit ordinal and for each ordinal number $\beta<\lambda$, $\forall k>0\forall\omega\cdot k\leq\alpha<\omega^2\,[(a\models p_k\wedge a\in\varphi_\alpha^\beta(\varnothing))\Rightarrow a\in\varphi_\alpha^{\omega\cdot k}(\varnothing)]$ holds. Fix $k>0$ and α such that $\omega\cdot k\leq\alpha<\omega^2$ and assume that $a\models p_k$ and $a\in\varphi_\alpha^\lambda(\varnothing)$. By definition of λ–th iteration of φ_α: $\varphi_\alpha^\lambda(\varnothing)=\bigcup_{\beta<\lambda}\varphi_\alpha^\beta(\varnothing)$, therefore there exists $\beta<\lambda$ such that $a\in\varphi_\alpha^\beta(\varnothing)$. Hence, by the induction hypothesis, $a\in\varphi_\alpha^{\omega\cdot k}(\varnothing)$.

[2] I would like to thank Michał Skrzypczak for showing me an idea how to present those formulae in a clearer way.

So it remains to prove the successor case:
Let $\beta = \gamma + 1$, then the induction hypothesis is of the form: $\forall k > 0 \forall \omega \cdot k \leq \alpha < \omega^2 [(a \models p_k \wedge a \in \varphi_\alpha^\gamma(\varnothing)) \Rightarrow a \in \varphi_\alpha^{\omega \cdot k}(\varnothing)]$. We need to show that $\forall k > 0 \forall \omega \cdot k \leq \alpha < \omega^2 [(a \models p_k \wedge a \in \varphi_\alpha^{\gamma+1}(\varnothing)) \Rightarrow a \in \varphi_\alpha^{\omega \cdot k}(\varnothing)]$.

Fix $k > 0$ and let $\alpha = \omega \cdot n + m$, for $n \geq k$ and $m \in \omega$. Let us assume that $a \models p_k$ and $a \in \varphi_\alpha^\beta(\varnothing)$. Since $\beta = \gamma + 1$ we have $a \in \varphi_\alpha(\varphi_\alpha^\gamma(\varnothing))$. By the definition of φ_α, one of the following cases must hold:

- $a \models \Box \bot$ – then $a \in \varphi_\alpha^0(\varnothing) \subseteq \varphi_\alpha^{\omega \cdot k}(\varnothing)$,
- $a \models C_l^n \wedge \Box C_l^n$ for some $l < k$ – since $a \models p_k$, and there exists t such that aRt and $t \in \varphi_\alpha^\gamma(\varnothing)$. Therefore $t \models C_l^n$ which implies $t \models p_{l+1}$. By the induction hypothesis, since $t \in \varphi_\alpha^\gamma(\varnothing)$ and $t \models p_{l+1}$, we know that $t \in \varphi_\alpha^{\omega \cdot (l+1)}(\varnothing)$. Because $\omega \cdot (l+1)$ is a limit ordinal, there exists $s \in \omega$ such that $t \in \varphi_\alpha^{\omega \cdot l + s}(\varnothing)$. Therefore $a \in \varphi_\alpha^{\omega \cdot l + s + 1}(\varnothing) \subseteq \varphi_\alpha^{\omega \cdot (l+1)}(\varnothing) \subseteq \varphi_\alpha^{\omega \cdot k}(\varnothing)$,
- $a \models C_{l+1}^n \wedge \Box C_l^n$ for some $l < k - 1$ – since $a \models p_k$, and for all t if aRt, then $t \in \varphi_\alpha^\gamma(\varnothing)$. Fix such t, then $t \models C_l^n$ and therefore $t \models p_{l+1}$, so by the induction hypothesis $t \in \varphi_\alpha^{\omega \cdot (l+1)}(\varnothing)$. Thus $a \in \varphi_\alpha^{\omega \cdot (l+1)+1}(\varnothing) \subseteq \varphi_\alpha^{\omega \cdot k}(\varnothing)$ since $l < k - 1$,
- In other cases, namely: $a \models \bigvee_{i=0}^{m-1} (\Box x \wedge \bigwedge_{j=0}^{i} \Box^j C_n^n \wedge \Box^{i+1} C_{n-1}^n)$, $a \models C_n^n$ which means $a \models \neg p_i$ for $i = 1, \ldots, n$, but this is a contradiction since $k \leq n$ and we assumed that $a \models p_k$.

This ends the proof of lemma 1. □

Lemma 1 shows us how the *fuses* work. If a point in which p_i is true is in the least fixpoint of φ_α, it has to be added relatively fast, that is in at most $\omega \cdot i$ steps. After that number of iterations the *fuse* p_i melts and no more points in which p_i is true may be added to the fixpoint. That is a global view on *fuses*, but we can also think about *fuses* locally. Let us consider the following example:

$$\begin{aligned}\varphi_{\omega \cdot 2+3} = &(\Diamond x \wedge C_0^2 \wedge \Box C_0^2) \vee (\Diamond x \wedge C_1^2 \wedge \Box C_1^2) \vee (\Box x \wedge C_1^2 \wedge \Box C_0^2) \vee \\ &\vee (\Box x \wedge C_2^2 \wedge \Box C_1^2) \vee (\Box x \wedge C_2^2 \wedge \Box C_2^2 \wedge \Box^2 C_1^2) \vee \\ &\vee (\Box x \wedge C_2^2 \wedge \Box C_2^2 \wedge \Box^2 C_2^2 \wedge \Box^3 C_1^2)\end{aligned}$$

We recall that $C_0^2 = p_1 \wedge p_2$, $C_1^2 = \neg p_1 \wedge p_2$ and $C_2^2 = \neg p_1 \wedge \neg p_2$. We can restrict our considerations to models that are trees. In every tree, in the first step all the leaves are added i.e. $\varphi_{\omega \cdot 2+3}^0(\varnothing) = [\![\Box \bot]\!]$. Now, there are four different types of points in the model: those that satisfy $p_1 \wedge p_2$, $p_1 \wedge \neg p_2$, $\neg p_1 \wedge p_2$ and $\neg p_1 \wedge \neg p_2$. If a point is of the first type and was added to the fixpoint in some iteration, then its parent will be added in the next iteration just in two cases – if it and all its children satisfy p_1 and p_2 ($\Diamond x \wedge C_0^2 \wedge \Box C_0^2$) or if all its children were already added to the fixpoint and it satisfies $\neg p_1$ and p_2 ($\Box x \wedge C_1^2 \wedge \Box C_0^2$). In the second case the *fuse* p_1 melted which says that ancestors of this point will never be added via $\Box x \wedge C_1^2 \wedge \Box C_0^2$. This melted *fuse* p_1 represents one use of the $\Box$. The second type of points are not very interesting as the parents of such points will never be added while iterating $\varphi_{\omega \cdot 2+3}$ – the second *fuse* is melted and the first is not while no disjunct in the formula satisfy such configuration. The third type of points is similar to the first one, but since they satisfy $\neg p_1$ and p_2 there is one less $\Box$ to use while trying to add their parents to the fixpoint. And again when $\Box x \wedge C_2^2 \wedge \Box C_1^2$ is used to add a parent of a point, a *fuse* p_2 melts and the parent added that way and its ancestors that are in the fixpoint do not satisfy p_2. In such case this was again the use of $\Box$ which might have let us pass the limit ordinal – namely $\omega \cdot 2$. The forth type of points – those which may be added to the fixpoint after melting the last *fuse* have nothing to do with *fuses* anymore – the formula is constructed in such way to ensure that there will be just $m - 1$ more essential iterations after melting the last *fuse*.

We now state the main theorem of the paper.

Theorem 1. *For every ordinal number α such that $\omega \leq \alpha < \omega^2$, in every model $\mathscr{M}$ and every valuation τ, $\varphi_\alpha^{\alpha+1}(\varnothing) = \varphi_\alpha^\alpha(\varnothing)$ holds.*

Proof. The inclusion $\supseteq$ is obvious by the monotonicity of φ_α in x. We show the other inclusion. Let $\alpha = \omega \cdot n + m$, for $n > 0$ and $m \in \omega$. Fix a model $\mathscr{M} = (M, R, V)$, valuation τ, and assume that $a \in \varphi_\alpha^{\alpha+1}(\varnothing)$. We show that $a \in \varphi_\alpha^\alpha(\varnothing)$.

If there exists $i \leq n$ such that $a \models p_i$, then, by lemma 1 we know that $a \in \varphi_\alpha^{\omega \cdot i}(\varnothing) \subseteq \varphi_\alpha^{\omega \cdot n + m}(\varnothing) = \varphi_\alpha^\alpha(\varnothing)$, since $a \in \varphi_\alpha^{\alpha+1}(\varnothing)$.

Let us now assume that for $i = 1, \ldots, n$, $a \models \neg p_i$ holds. Since $a \in \varphi_\alpha(\varphi_\alpha^\alpha(\varnothing))$, then by the definition of φ_α, $m > 0$ and one of the following cases must hold:

- $a \models \Box \bot$ – then trivially $a \in \varphi_\alpha^\alpha(\varnothing)$.
- $a \models \bigwedge_{j=0}^{i} \Box^j C_n^n \wedge \Box^{i+1} C_{n-1}^n$, for some $i = 0, \ldots, m-1$ and for every t such that aRt, $t \in \varphi_\alpha^\alpha(\varnothing)$.

  We proceed by induction on i to show that if $a \models \bigwedge_{j=0}^{i} \Box^j C_n^n \wedge \Box^{i+1} C_{n-1}^n$, then $a \in \varphi_\alpha^{\omega \cdot n + i + 1}(\varnothing)$. For the base step let us assume that $i = 0$. Then for all t such that aRt, $t \models C_{n-1}^n$ holds. Therefore $t \models p_n$ and by lemma 1, $t \in \varphi_\alpha^{\omega \cdot n}(\varnothing)$. Thus $a \in \varphi_\alpha^{\omega \cdot n + 1}(\varnothing)$.
  Suppose now that for $0 \leq i < k \leq m$ if $a \models \bigwedge_{j=0}^{i} \Box^j C_n^n \wedge \Box^{i+1} C_{n-1}^n$, then $a \in \varphi_\alpha^{\omega \cdot n + i + 1}(\varnothing)$. We show that for $i = k$ this implication holds as well. Suppose that $a \models \bigwedge_{j=0}^{k} \Box^j C_n^n \wedge \Box^{k+1} C_{n-1}^n$ then for every t such that aRt, $t \models \bigwedge_{j=0}^{k-1} \Box^j C_n^n \wedge \Box^k C_{n-1}^n$ holds. Therefore, by the induction hypothesis $t \in \varphi_\alpha^{\omega \cdot n + k}(\varnothing)$, and thus $a \in \varphi_\alpha^{\omega \cdot n + k + 1}(\varnothing)$.

  Hence, for every such case $a \in \varphi_\alpha^{\omega \cdot n + m}(\varnothing) = \varphi_\alpha^\alpha(\varnothing)$.
- In other cases, namely when $a \models \bigvee_{i=0}^{n-1} (\Diamond x \wedge C_i^n \wedge \Box C_i^n) \vee \bigvee_{i=0}^{n-2} (\Box x \wedge C_{i+1}^n \wedge \Box C_i^n)$ also $a \models C_i^{n+1}$ holds, for some $i = 1, \ldots n-1$. This means that $a \models p_n$ which is a contradiction, since we assumed that $a \models \neg p_n$.

This ends the proof. □

By theorem 1 for all ordinal numbers α such that $\omega \leq \alpha < \omega^2$ the least fixpoint of φ_α is reached in at most α steps. Now it is sufficient to show that for each formula φ_α there is a model such that φ_α reaches its least fixpoint in this model in exactly α steps.

Theorem 2. *For every ordinal number α such that $\omega \leq \alpha < \omega^2$ there is a model, in which φ_α reaches its least fixpoint in α steps.*

Proof. Let us recall the assignment of the trees to ordinal numbers sketched in the end of the previous section, where we show that $\mathscr{O}_x(\Box x) = \infty$. To 0 we assign just one point – the root of the tree. For a successor $\alpha + 1$ we add to the preexisting tree for α one more point, which is to be a new root, and attach to it simply the previous one. We change a limit step a bit: for a limit ordinal $\lambda = \omega \cdot n$ we also take a new point, which is to be a new root, but we attach to it just the roots of the trees $\omega \cdot (n-1)$, $\omega \cdot (n-1) + 1, \ldots$. Now we need to augment models assigned to the ordinal numbers $\alpha < \omega^2$ to models over $Prop = \{p_1, p_2, \ldots\}$. Let $n > 0$, $m \in \omega$, and $\alpha = \omega \cdot n + m$ and let $V_{\omega \cdot n + m}$ satisfy: $r_{\omega \cdot n + m} \in V(p_i)$ if and only if $i > n$, where $r_{\omega \cdot n + m}$ is the root of the corresponding model. The idea is to start with the model $\mathscr{M}_0$ whose root satisfies every p_i, for $i > 0$. When passing the successor step – the new root satisfies the same propositional variables as the previous one. Finally, for the limit step – if i is the least number for which all p_i are satisfied, in every point of models below, then the root of the limit model satisfies exactly p_j for $j > i$ (it does not satisfy p_i anymore). Violating the intuition, let us denote a model for an ordinal $\alpha < \omega^2$ by $\mathscr{M}_{\alpha+1}$. This construction works up to ω^2 and gives us models $\mathscr{M}_\alpha$, for each nonlimit ordinal α. By simple induction on α such that $\omega \leq \alpha < \omega^2$ we see that the formula φ_α reaches its least fixpoint $M_{\alpha+1} - \{r_{\alpha+1}\}$ in the model $\mathscr{M}_{\alpha+1}$ in exactly α steps. □

Corollary 1. *Let $\alpha < \omega^2$. Then $\mathscr{O}_x(\varphi_\alpha) = \alpha$ holds.*

The question whether there are formulae that reach their fixpoints in at least ω^2 iterations remains an open problem. Note that the formulae φ_α introduced in this paper are all basic modal formulae – we did not use the fixpoint operator in their construction. There may be some more complicated formulae which would allow us to control the number of iterations above ω^2. There also is a possibility to use *fuses* in the construction of such formulae, but this would require having a single formula that behaves as an infinite one–way counter at least in some big models[3] allowing us to count the number of uses of $\Box$ up to ω.

One can answer the following questions in order to extend the subject:

- Is there a basic modal logic formula which reaches its fixpoint in at least ω^2?
- Is there a modal μ calculus formula which reaches its fixpoint in at least ω^2?
- Is there a single formula behaving as an ω–counter at least in big enough models allowing us to count $\Box$ uses up to ω?

References

[1] G. Fontaine, *Continuous fragment of the μ–calculus*, Lecture Notes in Computer Science, vol. 5213/2008, Springer–Verlag, pp. 139–153.

[2] A. Arnold and D. Niwiński, *Rudiments of μ–Calculus*, Studies in Logic, Vol 146, North-Holland (2001).

[3] J. Bradfield and C. Stirling, Modal mu–calculi. In: P. Blackburn, J. van Benthem and F. Wolter (eds.), *The Handbook of Modal Logic* pp. 721-756. Elsevier (2006).

[3] Such models that there is a formula which reaches its fixpoint in those models in at least ω^2.

Proving fixed points

Hervé Grall
Research Team Ascola (EMN-INRIA, LINA)
Ecole des mines de Nantes, France
hgrall@mines-nantes.fr

Abstract

We propose a method to characterize the fixed points described in Tarski's theorem for complete lattices. The method is deductive: the least and greatest fixed points are "proved" in some inference system defined from deduction rules. We also apply the method to two other fixed point theorems, a generalization of Tarski's theorem to chain-complete posets and Bourbaki-Witt's theorem. Finally, we compare the method with the traditional iterative method resorting to ordinals and the original impredicative method used by Tarski.

We are interested in fixed points of maps defined over partially ordered sets (abbreviated as posets). Consider Tarski's fixed point theorem. Asserting the existence of fixed points under certain conditions, it is proved according to one of these two methods. In the *impredicative method*, the fixed points are characterized by a property (expressing extremality) using a quantification over a domain that includes the fixed point itself. In the *iterative method*, the fixed points are iteratively computed by using a transfinite induction. The impredicative method corresponds to a logical specification that is essentially not constructive[1]. As for the iterative method, it seems to be more constructive, in that it corresponds to an iteration. However, first, it assumes the machinery of ordinals and second, it therefore requires a specific computation for limit ordinals, the next value being then computed from an infinite set of preceding values.

Is there a proof method that not only is not impredicative, but also does not resort to ordinals? In this paper, we positively answer by proposing an alternative method, where fixed points are (inductively) *proved* in inference systems: this is a deductive method. It corresponds to a proof construction, using forward chaining for deduction rules, as used in logic programming, for instance in Datalog, a query language for deductive databases.

The paper is organized as follows. After recalling Tarski's theorem, the first section deals with inference systems, and their interpretations, either inductive or coinductive. In the second section, we introduce the deductive method, and apply it to different fixed point theorems: Tarski's theorem, its generalization to chain-complete posets and Bourbaki-Witt's theorem. Finally, we compare the deductive method with the iterative and impredicative methods. Note that the proofs presented here are just sketched: an extended version, available online[2], provides the details.

1 Induction and Coinduction for Inference Systems

Generally speaking, following Aczel's classical presentation [1], an *inference system* over a set $\mathscr{U}$ of judgments is a set of deduction rules. A *deduction rule* is an ordered pair (A,c), where $A \subseteq \mathscr{U}$ is the set of *premises* or *antecedents* and $c \in \mathscr{U}$ is the *conclusion*. A rule is usually written as follows:

$$\frac{A}{c}\,.$$

Its intuitive interpretation is that the judgment c can be deduced from the set of judgments A.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 41-46

[1] In a common sense: we do not specifically refer here to constructive mathematics.

[2] Downloadable from `http://hal.archives-ouvertes.fr/`.

Fixed Point Approach. The first way to assign a definitional meaning to an inference system is to consider the fixed points of the associated inference operator.

Indeed there exists a canonical Galois connection between the set of inference systems over $\mathscr{U}$ ordered by inclusion and the set of isotone (order-preserving) maps from $2^{\mathscr{U}}$ to $2^{\mathscr{U}}$ ordered point-wise. If Φ is an inference system over $\mathscr{U}$, then the associated operator $\varphi : 2^{\mathscr{U}} \to 2^{\mathscr{U}}$ is defined as follows:

$$\varphi(S) = \{c \in \mathscr{U} \mid \exists A \subseteq S.\ (A,c) \in \Phi\}.$$

The application of φ to S gives the conclusions that can be inferred in one step from S by using the inference rules: φ is thus called the *inference operator* associated to the system Φ. For instance, if S is the empty set, then $\varphi(S)$ is the set of the *axioms* of the inference system, in other words the conclusions of the rules without premises. Conversely, given an isotone operator $\varphi : 2^{\mathscr{U}} \to 2^{\mathscr{U}}$, the inference system containing all the rules (A,c), with $c \in \varphi(A)$, and only these rules, belongs to the inverse image of φ: this is the greatest element of the inverse image of φ.

Let Φ be an inference system and φ its associated inference operator. Since the powerset $2^{\mathscr{U}}$ is a complete lattice, by applying Tarski's fixed point theorem [9, p. 286], we obtain that the inference operator φ possesses both a *least fixed point* and a *greatest fixed point*.

We recall here Tarski's fixed point theorem, with two proof sketches following the impredicative and the iterative methods respectively.

Theorem 1 (Tarski)**.** *Let $(\mathscr{E},\leq)$ be a complete lattice. Let $\eta : \mathscr{E} \to \mathscr{E}$ be an isotone map over $\mathscr{E}$. Then η has a least fixed point* $\operatorname{lfp}\eta$ *and a greatest fixed point* $\operatorname{gfp}\eta$.

Impredicative method. The proof uses the following characterization:

$$\operatorname{lfp}\eta \;=\; \wedge\{x \in \mathscr{E} \mid \eta(x) \leq x\} \quad \text{and} \quad \operatorname{gfp}\eta \;=\; \vee\{x \in \mathscr{E} \mid x \leq \eta(x)\},$$

defining the least and greatest fixed points as the smallest η-closed set and the greatest η-consistent set respectively. It can be found in original Tarski's paper [9]. □

Iterative method. The proof uses the following characterization:

$$\operatorname{lfp}\eta \;=\; \bigvee_\alpha \Delta_\alpha(\eta) \quad \text{and} \quad \operatorname{gfp}\eta \;=\; \bigwedge_\alpha \nabla_\alpha(\eta).$$

where the sequences $(\Delta_\alpha(\eta))_\alpha$ and $(\nabla_\alpha(\eta))_\alpha$ are defined over ordinals as follows:

$$\Delta_\alpha(\eta) \;=\; \eta\Big(\bigvee_{\beta|\beta<\alpha} \Delta_\beta(\eta)\Big) \quad \text{and} \quad \nabla_\alpha(\eta) \;=\; \eta\Big(\bigwedge_{\beta|\beta<\alpha} \nabla_\beta(\eta)\Big).$$

It can be found in Cousot's article [5], for instance, with a slight variant in the definition of the sequences. □

Deductive Method. In contrast with the fixed point approach, the deductive method starts from the *proofs* admissible in an inference system. These proofs are represented as trees, called *proof trees*. These are trees whose nodes are labeled with judgments in $\mathscr{U}$ and such that for all nodes n, the label c of n and the labels A of the sons of n correspond to an inference rule (A,c) in Φ. The conclusion of a proof is the label of its root node. A proof d is *well-founded* if it has no infinite branch; d is *ill-founded* otherwise. Note that an ill-founded proof is always infinite. A well-founded proof is finite if and only if it only uses rules with a finite set of premises.

In the deductive method, the *inductive interpretation* of the inference system Φ is the set $\Delta(\Phi)$ of the conclusions of the well-founded proofs, while the *coinductive interpretation* is the set $\nabla(\Phi)$ of the conclusions of all the proofs, ill-founded or well-founded. The following theorem shows that the interpretations defined using fixed points and using proofs coincide.

Theorem 2 (Inductive and coinductive interpretations)**.** *Let Φ be an inference system and φ the associated inference operator. Then:*

$$\operatorname{lfp}\varphi \;=\; \Delta(\Phi) \quad \textit{and} \quad \operatorname{gfp}\varphi \;=\; \nabla(\Phi).$$

Proof. It is easy to show that $\Delta(\Phi)$ and $\nabla(\Phi)$ are a η-closed set and a η-consistent set respectively. In the reverse direction, given a η-closed set, an induction over well-founded proofs shows that any conclusion of a well-founded proof belongs to the η-closed set; given a η-consistent set, a system of guarded recursive equations over proof trees can be defined, generating as solution a valuation mapping each judgment in the η-consistent set to a proof tree with conclusion this judgment. By Tarski's theorem, we deduce $\operatorname{lfp}\varphi = \Delta(\Phi)$ and $\nabla(\Phi) = \operatorname{gfp}\varphi$. □

Details for the proof can be found in Leroy and author's article [7, Th. 1], where some bibliographic references are also given.

2 Generalization of the Deductive Method

Theorem 2 asserts that the least and greatest fixed points of an isotone map over a powerset, a particular complete lattice, can also be defined as the inductive and coinductive interpretations of an inference system. We first generalize to any complete lattice, and then to more powerful fixed point theorems.

Tarski's Theorem Revisited. Given an isotone map η over a complete lattice $(\mathscr{E},\leq)$, how can we define an inference system Φ over $\mathscr{E}$, or equivalently an inference operator $\varphi : 2^{\mathscr{E}} \to 2^{\mathscr{E}}$, whose inductive and coinductive interpretations produce the least and greatest fixed points of η?

A first attempt, defining $\varphi(S)$ as the image of S with η, trivially fails, since the least fixed point would be the empty set. However, a well-known result [4, Th. I.5.3] allows a complete lattice to be embedded in its powerset. Indeed, let $\gamma : 2^{\mathscr{E}} \to \{{\leq} x \mid x \in \mathscr{E}\}$ be the closure operator from the powerset of $\mathscr{E}$ to the set $\{{\leq} x \mid x \in \mathscr{E}\}$ of principal order ideals defined as follows: $\gamma(S) = {\leq}(\vee S)$[3]. Let $\delta : \{{\leq} x \mid x \in \mathscr{E}\} \to 2^{\mathscr{E}}$ be its adjoint embedding: $\delta({\leq} x) = {\leq} x$. Now, the isomorphism ι from $\mathscr{E}$ to $\{{\leq} x \mid x \in \mathscr{E}\}$ is defined as follows: $\iota(x) = {\leq} x$. Finally, thanks to the embedding via a closure operator, we can associate to the map η the operator $\varphi : 2^{\mathscr{E}} \to 2^{\mathscr{E}}$, equal to $\delta \circ \iota \circ \eta \circ \iota^{-1} \circ \gamma$: we have $\varphi(S) = {\leq}\eta(\vee S)$. It turns out that this operator has the intended properties with respect to fixed points. It suffices to use the associated inference system to get the following theorem, which we prove using the deductive method.

Theorem 3 (Tarski revisited)**.** *Let $(\mathscr{E},\leq)$ be a complete lattice. Let $\eta : \mathscr{E} \to \mathscr{E}$ be an isotone map over $\mathscr{E}$. Consider the inference system Φ over $\mathscr{E}$ containing all the rules, and only these rules, of the following form:*

$$\frac{S}{s} \quad \big(S \subseteq \mathscr{E}, s \leq \eta(\vee S)\big).$$

Then:

$$\leq(\operatorname{lfp}\eta) \;=\; \Delta(\Phi) \quad \textit{and} \quad \leq(\operatorname{gfp}\eta) \;=\; \nabla(\Phi).$$

Proof. It is easy to show that the inference operator φ associated to Φ is equal to $\delta \circ \iota \circ \eta \circ \iota^{-1} \circ \gamma$, with the preceding notation. As $\leq(\operatorname{lfp}\eta) = \operatorname{lfp}\varphi$ and $\leq(\operatorname{gfp}\eta) = \operatorname{gfp}\varphi$, we can conclude by applying Theorem 2. □

[3] Here, and in the following, given $x \in \mathscr{E}$, we denote by ${\leq} x$ the principal order ideal $\{y \in \mathscr{E} \mid y \leq x\}$. Likewise, if $X \subseteq \mathscr{E}$, we denote by ${\leq} X$ the union $\bigcup_{x \in X}({\leq} x)$.

Two Other Fixed Point Theorems. The deductive characterization of fixed points is still pertinent when we consider two other well-known fixed point theorems, where the assumptions for the poset and the map are weakened.

A poset is *chain-complete* if any chain, including the empty chain, has a least upper bound[4]. Tarski's theorem can be extended to chain-complete posets, as shown for instance by Markowsky [8, Th. 9]. We again resort to the preceding inference system to characterize the least fixed point, with two restrictions: we only consider chains as premises and greatest judgments as conclusions.

Theorem 4 (Fixed point theorem for chain-complete poset)**.** *Let $(\mathscr{E},\leq)$ be a chain-complete poset and $\eta : \mathscr{E} \to \mathscr{E}$ an isotone map. Let Φ be the inference system over $\mathscr{E}$ containing all the rules, and only these rules, of the following form:*

$$\frac{C}{\eta(\vee C)} \quad \big(C \subseteq \mathscr{E}, C \text{ chain}\big).$$

Then η has a least fixed point $\operatorname{lfp}\eta$ *satisfying:*

$$\leq(\operatorname{lfp}\eta) \quad = \quad \leq\Delta(\Phi).$$

Proof. We first show that $\Delta(\Phi)$ is a chain, precisely that for any x_1 and any x_2 in $\Delta(\Phi)$, we have $x_1 \leq x_2$ or $x_2 \leq x_1$. We proceed by induction over well-founded proofs. Consider a well-founded proof ended with the rule $(C_1, \eta(\vee C_1))$. Assume as inductive hypothesis that for any $y_1 \in C_1$ and any $y_2 \in \Delta(\Phi)$, we have $y_1 \leq y_2$ or $y_2 \leq y_1$. Let x_2 be in $\Delta(\Phi)$. There exists a chain C_2 included in $\Delta(\Phi)$ such that $x_2 = \eta(\vee C_2)$. There are two cases.

First, assume $\exists y_1 \in C_1 \,.\, \forall y_2 \in C_2 \,.\, y_2 \leq y_1$. We therefore have, for some y_1 in C_1 and for all y_2 in C_2: $y_2 \leq y_1$, $y_2 \leq \vee C_1$ and $\vee C_2 \leq \vee C_1$. We deduce by monotony $x_2 \leq \eta(\vee C_1)$.

Second, assume $\forall y_1 \in C_1 \,.\, \exists y_2 \in C_2 \,.\, \neg(y_2 \leq y_1)$. We therefore have, for all y_1 in C_1 and some dependent point y_2 in C_2: $y_1 \leq y_2$ (inductive hypothesis), $y_1 \leq \vee C_2$ and $\vee C_1 \leq \vee C_2$. We deduce by monotony $\eta(\vee C_1) \leq x_2$.

Since $\Delta(\Phi)$ is a chain, $\big(\Delta(\Phi), \eta(\vee\Delta(\Phi))\big)$ is a rule. We have $\eta(\vee\Delta(\Phi)) \in \Delta(\Phi)$, hence $\eta(\vee\Delta(\Phi)) \leq \vee\Delta(\Phi)$ and then $\eta^2(\vee\Delta(\Phi)) \leq \eta(\vee\Delta(\Phi))$ by monotony. Assume x is η-closed: $\eta(x) \leq x$. It is easy to show by induction over well-founded proofs that x is an upper bound of $\Delta(\Phi)$. We deduce that first $\vee\Delta(\Phi) \leq \eta(\vee\Delta(\Phi))$ and second $\vee\Delta(\Phi)$ is the least fixed point of η. Finally $\leq(\operatorname{lfp}\eta) = \leq\Delta(\Phi)$ since $\vee\Delta(\Phi) \in \Delta(\Phi)$. □

Actually, as shown by Markowsky [8, Th. 9], the preceding theorem can be considered as a corollary of Bourbaki-Witt's theorem [3]. We also give a proof of this theorem by using the same inference system as in Theorem 4.

Theorem 5 (Bourbaki-Witt)**.** *Let $\mathscr{E}$ be a chain-complete poset and $\eta : \mathscr{E} \to \mathscr{E}$ an expansive*[5] *map. Let Φ be the inference system over $\mathscr{E}$ containing all the rules, and only these rules, of the following form:*

$$\frac{C}{\eta(\vee C)} \quad \big(C \subseteq \mathscr{E}, C \text{ chain}\big).$$

Then η has a fixed point $\operatorname{fp}\eta$ *satisfying:*

$$\leq(\operatorname{fp}\eta) \quad = \quad \leq\Delta(\Phi).$$

[4] A chain-complete poset has therefore a bottom element, $\vee\emptyset$.

[5] A map $\eta : \mathscr{E} \to \mathscr{E}$ is *expansive* (also inflationary, progressive) if any point is η-consistent: $\forall x \in \mathscr{E} \,.\, x \leq \eta(x)$.

Proof. We show that $\Delta(\Phi)$ is a chain, which allows to conclude. Indeed, if $\Delta(\Phi)$ is a chain, the inference rule $\big(\Delta(\Phi), \eta(\vee\Delta(\Phi))\big)$ belongs to Φ. From $\eta(\vee\Delta(\Phi)) \in \Delta(\Phi)$, we deduce $\eta(\vee\Delta(\Phi)) \leq \vee\Delta(\Phi)$. Since η is expansive, we deduce that $\vee\Delta(\Phi)$ is a fixpoint of η. Moreover, $\vee\Delta(\Phi)$ belongs to $\Delta(\Phi)$. Therefore $\leq\Delta(\Phi) = \leq(\vee\Delta(\Phi))$.

To show that $\Delta(\Phi)$ is a chain, we introduce the notion of useful proofs. A well-founded proof, say ended by the rule $(C, \eta(\vee C))$, is said *useful* if for any well-founded proof, say ended by the rule $(D, \eta(\vee D))$, we have:

$$(\vee D < \vee C) \Rightarrow (\eta(\vee D) \leq \vee C).$$

Two crucial properties about usefulness can be asserted. First, for any useful proof ended by $(C, \eta(\vee C))$ and any well-founded proof ended by $(D, \eta(\vee D))$, we have either $\eta(\vee C) = \eta(\vee D)$, $\eta(\vee C) \leq \vee D$ or $\eta(\vee D) \leq \vee C$. Second, all well-founded proofs are useful.

Thanks to these properties, it is easy to conclude that $\Delta(\Phi)$ is a chain. Indeed, let x_1 and x_2 be in $\Delta(\Phi)$. There exists two well-founded proofs, respectively ended by $(C_1, \eta(\vee C_1))$ and $(C_2, \eta(\vee C_2))$, such that $x_1 = \eta(\vee C_1)$ and $x_2 = \eta(\vee C_2)$. Since the former proof is useful, we have by applying the first property either $x_1 = x_2$, $x_1 \leq \vee C_2$ or $x_2 \leq \vee C_1$. Since η is expansive, we deduce $x_1 \leq x_2$ or $x_2 \leq x_1$.

The two properties about usefulness can be proved by induction over well-founded proofs. The arguments used are standard, since the notion of useful proofs comes from the notion of *extreme points*, used in the proof of the Bourbaki-Witt's theorem following the impredicative method: see for instance the proof in Lang's book [6, pp. 881–884]. □

Comparison of the Methods. Tarski's theorem and its two extensions to chain-complete posets are usually proved with the iterative method or the impredicative method, two methods that are not clearly connected. We now suggest that the deductive method allows these two methods to be connected.

First, to each inference system Φ used in Theorems 3, 4 and 5, we can associate an inference operator φ over $2^{\mathscr{E}}$. Thus, the inductive interpretation $\Delta(\Phi)$ of Φ can also be defined as the smallest φ-closed set, as expressed in Theorems 1 and 2. It turns out that the definition of a φ-closed set is very akin to the definition of an admissible set, as found in the standard proof of Bourbaki-Witt's theorem following the impredicative method [6, pp. 881–884].

Second, the deductive method can be considered as an abstraction of the iterative method: it abstracts away from the iterative process involved in proof construction. The following proposition precisely describes their relationship in the case of Tarski's theorem. The *height* of a well-founded proof tree d is defined as follows: it is the least ordinal greater than the height of each immediate proof sub-tree of d. Given an inference system Φ over $\mathscr{E}$ and an ordinal α, we say that $x \in \mathscr{E}$ has complexity α if there is a well-founded proof of x with height less or equal to α. We denote by $\Delta_\alpha(\Phi)$ the set of all x with complexity α.

Proposition 1. *Let $(\mathscr{E}, \leq)$ be a complete lattice and $\eta : \mathscr{E} \to \mathscr{E}$ an isotone map over $\mathscr{E}$. Consider the transfinite sequence $(\Delta_\alpha(\eta))_\alpha$ defined as follows:*

$$\Delta_\alpha(\eta) \quad = \quad \eta(\bigvee_{\beta|\beta<\alpha} \Delta_\beta(\eta)).$$

Consider the inference system Φ containing all the rules, and only these rules, of the following form:

$$\frac{S}{s} \quad \big(S \subseteq \mathscr{E}, s \leq \eta(\vee S)\big).$$

Then, for any ordinal α:

$$\leq\Delta_\alpha(\eta) = \Delta_\alpha(\Phi).$$

Proof. By transfinite induction. □

An analogous proposition holds for the two other fixed point theorems.

3 Conclusion

We have presented a method to characterize the fixed points described in fixed point theorems for complete lattices and chain-complete posets. The method is deductive: the fixed points are "proved" in some inference system. The techniques used in the proofs of these theorems is consistent with the method: they are based on induction over well-founded proofs. Finally, we have sketched a comparison between the deductive method and the two traditional methods, impredicative and iterative. In brief, given an inference system, the impredicative method essentially corresponds to the characterization à la Tarski using the inference operator associated to the inference system, whereas the iterative method corresponds to an iterative construction of the well-founded proofs in the inference system. The connection that we have suggested deserves a further exploration, which we reserve to future work.

We have already experienced the deductive method in a work about coinductive operational semantics [7]. It turns out that the method is well-suited to the proof assistant Coq, which uses a calculus of inductive and coinductive constructions, an extension of type theory. The fixed points were expressed through inference systems and defined over powerset lattices. A motivation of the present work was to extend the method to fixed points defined in Coq over complete lattices or chain-complete posets. The iterative method resorts to ordinals, which are rarely used in Coq. Indeed, either they require to encode set theory, which is expensive, or they are implemented as constructive ordinals, which is restrictive. As for the impredicative method, it does not really fit with the calculus of inductive and coinductive constructions. Thus, the deductive method is a good candidate. A major problem to be solved is the use of classical logic. First, since the fixed points defined may have a non-terminating behavior, for instance when the fixed point is a function possibly divergent, adding classical logic axioms to the constructive logic of Coq is needed. In the same way, in the preceding proofs of the fixed point theorems, we have used the law of excluded middle. Second, as explained in the recent proposal of Bertot and Komendantsky [2], the addition of classical logic axioms should allow not only to reason about a fixed point with a terminating or non-terminating behavior, but also to extract a program computing the fixed point from the proof that the fixed point satisfies its specification, following the Curry-Howard correspondence.

References

[1] Peter Aczel. An introduction to inductive definitions. In Jon Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.7, pages 739–782. North-Holland, 1977.

[2] Yves Bertot and Vladimir Komendantsky. Fixed point semantics and partial recursion in Coq. In *Proceedings of the 10th ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP '08)*, pages 89–96. ACM Press, 2008.

[3] Nicolas Bourbaki. *Théorie des ensembles : Fascicule de résultats*. Hermann, 1939.

[4] Stanley Burris and Hanamantagouda Sankappanavar. *A Course in Universal Algebra*. Number 78 in Graduate Texts in Mathematics. Springer-Verlag, 1981.

[5] Patrick Cousot and Radhia Cousot. Constructive versions of tarski's fixed point theorems. *Pacific Journal of Mathematics*, 82(1):43–57, 1979.

[6] Serge Lang. *Algebra, Revised Third Edition*. Graduate Texts in Mathematics. Springer-Verlag, 2002.

[7] Xavier Leroy and Hervé Grall. Coinductive big-step operational semantics. *Information and Computation*, 207(2):284–304, 2009.

[8] George Markowsky. Chain-complete posets and directed sets with applications. *Algebra Universalis*, 6:53–68, 1976.

[9] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.

Characterizing Recursive Programs Up To Bisimilarity

Paul Blain Levy
University of Birmingham

Abstract

A recursive program is determined, up to bisimilarity, by the operation of the recursion body on arbitrary processes, of which it is a fixpoint. The traditional proof of this fact uses Howe's method, but that does not tell us how the fixpoint is obtained.

In this paper, we show that the fixpoint may be obtained by a least fixpoint procedure iterated through the hierarchy of countable ordinals, using Groote and Vaandrager's notion of nested simulation.

1 Introduction

Recursion is an important programming language feature that provides a fixpoint of an endofunction. But an endofunction may have many fixpoints, so an important question in semantics is to determine which is the one calculated by recursion. For example, for both the may-testing and must-testing preorders, recursion calculates the *least* pre-fixed point. (In the case of must-testing, we must assume the calculus uses erratic rather than ambiguous nondeterminism, see e.g. [Las98].)

What if we work modulo bisimilarity? We provide a characterization of the fixpoint calculated by recursion as follows. First calculate the least pre-fixed point up to similarity. Within this equivalence class, calculate the least pre-fixed point up to 2-nested similarity. Iterate this procedure through all the countable ordinals and it converges on a single point: the "nesting fixpoint". This is the one that recursion calculates.

In Sect. 2 we introduce the general notions of nested simulation and nesting fixpoints. We illustrate them in Sect. 3 with the process calculus CCS.

Notation

- We write ω_1 for the least uncountable ordinal.
- For any sets X, Y, Z and relations $\mathscr{R} \subseteq X \times Y$ and $\mathscr{R}' \subseteq Y \times Z$, we write $\mathscr{R};\mathscr{R}'$ for the composite.
- For any sets X and Y, we write $X \rightharpoonup Y$ for the set of partial functions from X to Y with finite domain.
- For any set X with preorder $\leqslant$, and any $\leqslant \cap \geqslant$-equivalence class U, we write

$$\downarrow^{\leqslant}(U) \stackrel{\text{def}}{=} \{x \in X \mid \exists y \in X.\ x \leqslant y\} = \{x \in X \mid \forall y \in X.\ x \leqslant y\}$$

2 Transition systems and ω_1-nested preorders

We first recall the basic notions of transition systems.

Definition 1. *Let* Act *be a set of* actions*, and let* $\mathscr{S} = (X, \rightarrow)$ *be an* Act-labelled transition system*, i.e. a set* X *together with a relation* $\rightarrow \subseteq X \times \mathsf{Act} \times X$.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 47-52

1. *For each $x \in X$ and $a \in \mathsf{Act}$, we write*

$$\mathsf{succ}_a(P) \stackrel{\text{def}}{=} \{Q \in \mathsf{Prog} \mid P \xrightarrow{a} Q\}$$

If this set is always countable, $\mathscr{S}$ is said to be image-countable.

2. *A relation $\mathscr{R} \subseteq X \times X$ is*

   - *a* simulation *on $\mathscr{S}$ when for all $(x,x') \in \mathscr{R}$ and $a \in \mathsf{Act}$, if $y \in \mathsf{succ}_a(x)$ then there exists $y' \in \mathsf{succ}_a(x')$ such that $(x',y') \in \mathscr{R}$.*
   - *a* bisimulation *on $\mathscr{S}$ when both $\mathscr{R}$ and its converse are simulations.*

3. *The greatest bisimulation is called* bisimilarity. *It is an equivalence relation and written $\approx$.*

4. *Let $\leqslant$ be a preorder on X. A relation $\mathscr{R} \subseteq X \times X$ is*

   - *a* simulation up to $\leqslant$ on the right *when for all $(x,x') \in \mathscr{R}$ and $a \in \mathsf{Act}$, if $y \in \mathsf{succ}_a(x)$ then there exists $y' \in \mathsf{succ}_a(x')$ such that $(x',y') \in (\mathscr{R};\leqslant)$*
   - *a* simulation up to $\leqslant$ *when for all $(x,x') \in \mathscr{R}$ and $a \in \mathsf{Act}$, if $y \in \mathsf{succ}_a(x)$ then there exists $y' \in \mathsf{succ}_a(x')$ such that $(x',y') \in (\leqslant;\mathscr{R};\leqslant)$*

Definition 2. *[GV92] Let* Act *be a set, and let $\mathscr{S} = (X,\to)$ be an* Act*-labelled transition system. For each ordinal α, we shall define a preorder $\lesssim_\alpha$, known as* α-nested similarity*, with the property that any simulation contained in $\gtrsim_\alpha$ is also contained in $\lesssim_\alpha$. We define $\lesssim_\alpha$ to be*

($\alpha = \beta + 1$) *the greatest simulation contained in $\lesssim_\beta$, or equivalently the greatest simulation contained in $\lesssim_\beta \cap \gtrsim_\beta$*

(α a limit ordinal) *the intersection of $\lesssim_\beta$ over all $\beta < \alpha$.*

Lemma 1. *[GV92] Let* Act *be a set, and let $(X,\to)$ be an* Act*-labelled transition system.*

1. *Let β be an ordinal. If $\mathscr{R}$ is a simulation up to $\lesssim_{\beta+1}$ contained in $\gtrsim_\beta$, then $\mathscr{R}$ is contained in $\lesssim_{\beta+1}$.*
2. *$\lesssim_\alpha$ contains bisimilarity, for each ordinal α.*
3. *If $(X,\to)$ is image-countable, then $\lesssim_{\omega_1}$ is bisimilarity.*

We are thus led to the following abstract notion.

Definition 3. *Let X be a set. An* ω_1-nested preorder *on X is a sequence of preorders $(\leqslant_\alpha)_{\alpha \leqslant \omega_1}$ such that*

- *$(\leqslant_{\alpha+1}) \subseteq (\leqslant_\alpha) \cap (\geqslant_\alpha)$, for each $\alpha < \omega_1$*
- *$(\leqslant_\gamma) = \bigcap_{\alpha<\gamma}(\leqslant_\alpha)$, for each limit ordinal $\gamma \leqslant \omega_1$*

It follows that

- $(\leqslant_0)$ is the indiscrete relation
- $\alpha \leqslant \beta \leqslant \omega_1$ implies $(\leqslant_\beta) \subseteq (\leqslant_\alpha)$
- $\alpha < \beta \leqslant \omega_1$ implies $(\leqslant_\beta) \subseteq (\geqslant_\alpha)$

- $(\leqslant_\alpha)$ is an equivalence relation, for each limit ordinal $\alpha \leqslant \omega_1$.

In particular, $\leqslant_{\omega_1}$ is an equivalence relation, which we write $\equiv$.

Definition 4. *Let* Act *be a set, and let $\mathscr{S} = (X, \to)$ be an* Act*-labelled transition system. We write $A(\mathscr{S})$ for the ω_1-nested preordered set $(X, (\lesssim_\alpha)_{\alpha \leqslant \omega_1})$.*

Definition 5. *Let $A = (X, (\leqslant_\alpha)_{\alpha \leqslant \omega_1}$ and $B = (Y, (\leqslant_\alpha))_{\alpha \leqslant \omega_1}$ be ω_1-nested preordered sets. A* monotone function $A \xrightarrow{f} B$ *is a function $X \xrightarrow{f} Y$ such that, for every $\alpha \leqslant \omega_1$ (or equivalently: every successor ordinal $\alpha < \omega_1$), if $x \subseteq^\alpha_A x'$ then $f(x) \subseteq^\alpha_B f(x')$.*

Now we come to our key definition.

Definition 6. *Let $A = (X, (\leqslant_\alpha)_{\alpha \leqslant \omega_1}$ be an ω_1-nested preordered set, and let f be a monotone endofunction on A. We shall define a decreasing sequence of subsets $(U^f_\alpha)_{\alpha \leqslant \omega_1}$ of X such that U^f_α either is empty or satisfies the following conditions:*

- *U^f_α is an equivalence class of $\leqslant_\alpha \cap \geqslant_\alpha$*
- *f restricts to an endofunction on U^f_α and hence on $\downarrow^{\leqslant_\alpha}(U^f_\alpha)$*
- *if $x \in \downarrow^{\leqslant_\alpha}(U^f_\alpha)$ and $f(x) \leqslant_\alpha x$ then $x \in U^f_\alpha$.*

We define U^f_α to be

($\alpha = \beta + 1$) *the set of $\leqslant_\alpha$-least elements of*

$$\{x \in U^f_\beta \mid f(x) \leqslant_\alpha x\} = \{x \in \downarrow^{\leqslant_\beta}(U^f_\beta) \mid f(x) \leqslant_\alpha x\}$$

(α a limit ordinal) *the intersection of U^f_β over all $\beta < \alpha$.*

The elements of $U^f_{\omega_1}$ are called nesting fixpoints *of f.*

Note that nesting fixpoints are fixpoints up to $\equiv$ and unique up to $\equiv$. But some monotone endofunctions f do not have a nesting fixpoint—i.e. $U^f_{\omega_1}$ is empty.

3 CCS and Bisimilarity

Our thesis is that a recursive program in a transition system $\mathscr{S}$ is is a nesting fixpoint of the monotone endofunction on $A(\mathscr{S})$ given by the recursion body. To illustrate this, we consider the calculus CCS [Mil89], over a fixed set Act of actions.

As CCS is untyped, a *context* Γ is merely a list of distinct identifers. The syntax is given inductively by the rules in Fig. 1. We write Prog for the set of *programs*, i.e. closed terms, which forms an Act-labelled transition system with transition relation $\to$ defined inductively by the rules in Fig. 2. This system is easily shown to be image-countable, and we call it CCS.

Our version of CCS includes parallel composition of any countable arity I, with synchronization described by a relation V saying when finitely many actions performed by the constituent processes may cause an action in the combined process. In [Mil89], Act is given by a disjoint union

$$\{a \mid a \in \Sigma\} \;\cup\; \{\overline{a} \mid a \in \Sigma\} \;\cup\; \{\tau\}$$

where Σ is a set of *synchronization actions*. The parallel composition, hiding and renaming operators provided there are subsumed by our parallel composition as follows.

$$\frac{\Gamma \vdash P}{\Gamma \vdash a.P}\ a \in \mathsf{Act} \qquad \frac{\Gamma \vdash P_i\ \ (\forall i \in I)}{\Gamma \vdash \sum_{i\in I} P_i}\ I \text{ countable} \qquad \frac{\Gamma, \mathtt{x} \vdash P}{\Gamma \vdash \mathtt{rec\ x.}\ P}$$

$$\frac{}{\Gamma \vdash \mathtt{x}}\ \mathtt{x} \in \Gamma \qquad \frac{\Gamma \vdash P_i\ \ (\forall i \in I)}{\Gamma \vdash \|^V_{i\in I} P_i}\ I \text{ countable}, V \subseteq (I \rightharpoonup_{\mathsf{fin}} \mathsf{Act}) \times \mathsf{Act}$$

Figure 1: Syntax of CCS

$$\frac{}{a.P \xrightarrow{a} P} \qquad \frac{P_{\hat{\imath}} \xrightarrow{a} Q}{\sum_{i\in I} P_i \xrightarrow{a} Q}\ \hat{\imath} \in I$$

$$\frac{P[\mathtt{rec\ x.}\ P/\mathtt{x}] \xrightarrow{a} Q}{\mathtt{rec\ x.}\ P \xrightarrow{a} Q} \qquad \frac{P_i \xrightarrow{b(i)} Q_i\ \ (\forall i \in \mathsf{dom}\ b)}{\|^V_{i\in I} P_i \xrightarrow{a} \|^V_{i\in I} \begin{cases} Q_i & (\text{if } i \in \mathsf{dom}\ b) \\ P_i & (\text{otherwise}) \end{cases}}\ (b,a) \in V$$

Figure 2: Operational Semantics (Transitions) of CCS

- We express $P \mid Q$ as $\|^V \{0 \mapsto P, 1 \mapsto Q\}$, with V given by

$$\{(\{0 \mapsto a, 1 \mapsto \overline{a}\}, \tau) \mid a \in \Sigma\} \ \cup\ \{(\{0 \mapsto \overline{a}, 1 \mapsto a\}, \tau) \mid a \in \Sigma\}$$
$$\cup\ \{(\{0 \mapsto a\}, a) \mid a \in \mathsf{Act}\} \ \cup\ \{(\{1 \mapsto a\}, a) \mid a \in \mathsf{Act}\}$$

- Let $f : \Sigma \to \Sigma$ be a function. We express $P[f]$ as $\|^V \{0 \mapsto P\}$, with V given by

$$\{(\{0 \mapsto a\}, f(a)) \mid a \in \Sigma\} \ \cup\ \{(\{0 \mapsto \overline{a}\}, \overline{f(a)}) \mid a \in \Sigma\} \ \cup\ \{(0 \mapsto \tau, \tau)\}$$

- Let $L \subseteq \Sigma$ be a subset. We express $P \backslash L$ as $\|^V \{0 \mapsto P\}$, with V given by

$$\{(\{0 \mapsto a\}, a) \mid a \in \Sigma \setminus L\} \ \cup\ \{(\{0 \mapsto \overline{a}\}, \overline{a}) \mid a \in \Sigma \setminus L\} \ \cup\ \{(0 \mapsto \tau, \tau)\}$$

The "synchronization algebras" of [WN95] are likewise expressible.

As explained in [Mil89], we could also incorporate into the language countably mutual recursion. We have not done so, but our results would go through without difficulty.

The following operations on programs are called the *basic operations*:

P	$\mapsto$	$a.P$	for any $a \in \mathsf{Act}$
$(P_i)_{i\in I}$	$\mapsto$	$\sum_{i\in I} P_i$	for any countable I
$(P_i)_{i\in I}$	$\mapsto$	$\|^V_{i\in I} P_i$	for any countable I and $V \subseteq (I \rightharpoonup_{\mathsf{fin}} \mathsf{Act}) \times \mathsf{Act}$

Proposition 1. *The basic operations preserve α-nested similarity, for every ordinal α, and hence preserves bisimilarity.*

Proof. Straightforward induction on α. Preservation of bisimilarity may also be proved directly. □

Lemma 2. *Let $\mathscr{R}$ be a simulation on CCS. Then the relation*

$$\{(M[P/\mathtt{x}], M[P'/\mathtt{x}]) \mid (P, P') \in \mathscr{R}, \mathtt{x} \vdash M\}$$

is also a simulation.

Proof. We want to show that if $M[P/\mathtt{x}] \xrightarrow{a} Q$, then for all P' such that $(P,P') \in \mathscr{R}$ we have $(R,R') \in \mathscr{R}$ and $\mathtt{x} \vdash N$ such that $Q = N[R/\mathtt{x}]$ and $M[P'/\mathtt{x}] \xrightarrow{a} N[R'/\mathtt{x}]$. We proceed by induction on $\rightarrow$. We omit the details. □

Proposition 2. *(Definable functions are monotone) Let $\mathtt{x} \vdash M$ be a term. Then the endofunction on* Prog

$$P \mapsto M[P/\mathtt{x}]$$

preserves α-similarity, for every ordinal α, and hence preserves bisimilarity.

Proof. By induction on α using Lemma 2. Preservation of bisimilarity also follows directly from Lemma 2. □

Lemma 3. *Let $\leqslant$ be a preorder on* Prog *that is a simulation and preserved by the basic operations. Let $\mathtt{x} \vdash M$ and $P \in$ Prog be such that $M[P/\mathtt{x}] \leqslant P$. Then the relation*

$$\{(N[\mathtt{rec\ x}.\ M/\mathtt{y}], N[P/\mathtt{y}]) \mid \mathtt{y} \vdash N\}$$

is a simulation up to $\leqslant$ on the right.

Proof. We want to show that if $N[\mathtt{rec\ x}.\ M/\mathtt{y}] \xrightarrow{a} Q$ then there exists $\mathtt{y} \vdash R$ and $Q' \in$ Prog such that $Q = R[\mathtt{rec\ x}.\ M/\mathtt{y}]$ and $N[P/\mathtt{y}] \xrightarrow{a} Q'$ and $R[N/\mathtt{y}] \leqslant Q'$. We proceed by induction on $\rightarrow$.

- Suppose that $N = \mathtt{y}$. Then $M[\mathtt{rec\ x}.\ M/\mathtt{x}] \xrightarrow{a} Q$ and applying the inductive hypothesis gives $\mathtt{y} \vdash R$ and $Q' \in$ Prog such that $Q = R[\mathtt{rec\ x}.\ M/\mathtt{y}]$ and $M[P/\mathtt{x}] \xrightarrow{a} Q'$ and $R[N/\mathtt{y}] \leqslant Q'$. Since $M[P/\mathtt{x}] \leqslant P$ and $\leqslant$ is a simulation we have $P \xrightarrow{a} Q''$ and $Q' \leqslant Q''$, giving $R[N/\mathtt{y}] \leqslant Q''$.
- The other cases are trivial.

□

Proposition 3. *Let $\mathtt{x} \vdash M$ be a term. Then* $\mathtt{rec\ x}.\ M$ *is a nesting fixpoint of the monotone endofunction $f : P \mapsto M[P/\mathtt{x}]$ on $A(CCS)$.*

Proof. We have to show that $\mathtt{rec\ x}.\ M$ is in U^f_α for each $\alpha \leqslant \omega_1$. The case where α is a limit ordinal is trivial, so suppose $\alpha = \beta + 1$. Since $\mathtt{rec\ x}.M \in U^f_\beta$ we have

$$\downarrow^{\leqslant\beta} (U^f_\beta) = \{P \in \mathsf{Prog} \mid P \lesssim_\beta \mathtt{rec\ x}.\ M\}$$

We need to show that $\mathtt{rec\ x}.\ M$ is an $\lesssim_{\beta+1}$-least element of

$$\{P \in \downarrow^{\leqslant\beta} (U^f_\beta) \mid f(P) \lesssim_\alpha P\} = \{P \in \mathsf{Prog} \mid P \lesssim_\beta \mathtt{rec\ x}.\ M \wedge M[P/\mathtt{x}] \lesssim_\alpha P\}$$

It is an element because $M[\mathtt{rec\ x}.\ M/\mathtt{x}] \eqsim \mathtt{rec\ x}.\ M$. Suppose P is another element. Then

$$\{(N[\mathtt{rec\ x}.\ M/\mathtt{x}], N[P/\mathtt{x}]) \mid \mathtt{x} \vdash N\}$$

is contained in $\gtrsim_\beta$ and, by Lemma 3, is a simulation up to $\lesssim_\alpha$ on the right. Lemma 1(1) tells us that it is contained in $\lesssim_\alpha$, so $\mathtt{rec\ x}.\ M \lesssim_\alpha P$ as required. □

Corollary 1. *Let $\mathtt{x} \vdash M, M'$ be terms such that $M[P/\mathtt{x}] \eqsim M'[P/\mathtt{x}]$ for all programs P. Then* $\mathtt{rec\ x}.\ M \eqsim \mathtt{rec\ x}.\ M'$.

This result may also be proved using Howe's method [How96, Lev06].

4 Conclusions and Further Work

The present paper was greatly inspired by the denotational semantics in [Ros04], where recursion is interpreted by a "reflected" fixpoint calculated in two steps.

The quotient of CCS by bisimilarity is an image-countable transition system $\mathscr{S}$ in which bisimilarity is discrete. (It can also be described as a final coalgebra [TR98].) Therefore nesting fixpoints in $A(\mathscr{S})$ are genuine fixpoints and unique. This almost provides a denotational semantics, except that some monotone endofunctions do not have a nesting fixpoint. Perhaps restricting to the *exploratory* functions of [LW09] would be fruitful, as these are all definable in a sufficiently rich calculus.

In [Abr91] a domain theoretic model is provided that captures bisimilarity between processes without divergences. For general processes it induces a more subtle preorder.

The results of this paper may be adapted to lower (i.e. divergence-insensitive) applicative bisimulation [Abr90] in nondeterministic λ-calculus. However, in this instance Howe's method is stronger because it shows applicative bisimilarity to be preserved not only by recursion (Corollary 1) but also by application.

References

[Abr90] S. Abramsky. The lazy λ-calculus. In *Research topics in Functional Programming*, pages 65–117. Addison Wesley, 1990.

[Abr91] S Abramsky. A domain equation for bisimulation. *Information and Computation*, 92(2), 1991.

[GV92] Jan Friso Groote and Frits Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

[How96] D J Howe. Proving congruence of bisimulation in functional programming languages. *Inf. and Comp.*, 124(2), 1996.

[Las98] S B Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Univ. of Aarhus, 1998.

[Lev06] P B Levy. Infinitary Howe's method. In *Proc., 8th Intl. Workshop on Coalgebraic Methods in Comp. Sci., Vienna*, volume 164(1) of *ENTCS*, 2006.

[LW09] Paul Blain Levy and Kidane Yemane Weldemariam. Exploratory functions on nondeterministic strategies, up to lower bisimilarity. *Electr. Notes Theor. Comput. Sci*, 249:357–375, 2009.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Ros04] A W Roscoe. Seeing beyond divergence. presented at BCS FACS meeting "25 Years of CSP", July 2004.

[TR98] Daniele Turi and Jan J. M. M. Rutten. On the foundations of final coalgebra semantics. *Mathematical Structures in Computer Science*, 8(5):481–540, 1998.

[WN95] G. Winskel and M. Nielsen. Models for concurrency. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*. Oxford University Press, 1995.

The Equivalence of Game and Denotational Semantics for the Probabilistic μ-Calculus

Matteo Mio
LFCS, School of Informatics, University of Edinburgh
M.Mio@sms.ed.ac.uk

1 Introduction

The modal μ-calculus L_μ [7] is a very expressive logic obtained by extending classical propositional modal logic with least and greatest fixed point operators. The logic L_μ has been extensively studied as it provides a very powerful tool for expressing properties of labeled transition systems [14]. Encodings of many important temporal logics such as LTL, CTL and CTL* into L_μ [1], provided evidence for the very high expressive power of the calculus. A precise expressivity result was given in [6], where the authors showed that every formula of monadic second order logic over transition systems which does not distinguish between bisimilar models is equivalent to a formula of L_μ. The logic L_μ has a simple denotational interpretation [14] and an elegant proof theory [15]. However it is often very difficult to intuitively grasp the denotational *meaning* of a L_μ formula as the nesting of fixed point operators can induce very complicated properties. To alleviate this problem, another complementary semantics for the logic L_μ, based on two player (parity) games, has been studied in [3, 14]. The two semantics have been proven to coincide and this allows us to pick the most convenient viewpoint when reasoning about the logic L_μ. One of the main properties of the games used to give semantics to L_μ formulae is the so called *positional determinacy* which asserts that both Players can play optimally in a given game-configuration without knowing the history of the previously played moves.

In the last decade, a lot of research has focused on the study of reactive systems that exhibit some kind of probabilistic behavior, and logics for expressing their properties. Segala systems [13] are a natural generalization of labeled transition systems to the probabilistic scenario. Given a countable set of labels L, a Segala System is a pair $\langle P, \{\xrightarrow{a}\}_{a\in L}\rangle$ where P is a countable set of states and, for each $a \in L$, $\xrightarrow{a} \subseteq P \times \mathscr{D}(P)$ is the a-accessibility relation, where $\mathscr{D}(P)$ is the set of probability distributions over P. The transition relation models the dynamics of the processes: $(p,d) \in \xrightarrow{a}$ means that the process p can perform the atomic action $a \in L$ and then behave like the process q with probability $d(q)$.
The probabilistic modal μ-calculus pL_μ, introduced in [12, 5, 2], is a generalization of L_μ designed for expressing properties of Segala systems. This logic was originally named *quantitative* μ-calculus, but since other μ-calculus-like logics, designed for expressing properties of non-probabilistic systems, have been given the same name (e.g. [4]), we adopt the *probabilistic* adjective.
The denotational semantics for the logic pL_μ of [12, 2], interprets every formula F as a map $[\![F]\!] : P \to [0,1]$, which assigns to each process p a *degree of truth*. Actually, in [5] three different possible denotational semantics for pL_μ (including the one of [12, 2]) have been proposed as there there is no, *a priori*, good reason to prefer one in favour of the others.
In [9, 10], the authors introduce a game semantics for the logic pL_μ. This semantics, given in term of two player stochastic (parity) games, is the natural generalization of the two player (non stochastic) game semantics for the logic L_μ; the key difference being that in the configuration $\langle p, \langle a\rangle F\rangle$ (respectively $\langle p, [a]F\rangle$) Player 1 (respectively Player 2) choses a a-successor of p, i.e. a distribution d such that $(p,d) \in \xrightarrow{a}$, and the next configuration $\langle q, F\rangle$ is then reached with probability $d(q)$. This semantics

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 53-59

allows one to interpret formulae as expressing, for each process p, the (limit) probability of the *property* specified by the formula to hold in the state p. This game semantics suggests a very clear *operational* interpretation for the logical connectives of the logic which, *a posteriori*, justifies the denotational interpretation of [12, 2].
In [10, 9], the authors proved the equivalence of the denotational and game semantics for pL_μ *only* for finite models. The proof crucially depends on positional determinacy which does not hold, in general, for the infinite pL_μ stochastic parity games generated by infinite models. The general result, i.e. the equivalence of the game and denotational semantics for arbitrary infinite models, has been left open.
In this workshop paper we show that the equivalence indeed holds for arbitrary infinite models, thus strengthening the connection between denotational and game semantics. Our contribution consists in adapting the technique introduced in [4], where the authors used it to prove a similar result for a μ-calculus-like logic designed to express quantitative properties of (non-probabilistic) labeled transition systems. While this is not a difficult adaption, the result seems worth noticing since the question has been open in literature since [9]. Moreover the differences between the games considered in [4] and pL_μ stochastic games, e.g. the fact that *Markov chains* are the outcomes of the games rather than just infinite paths, make this result not immediate from [4].

The rest of the paper is organized as follows: in section 2 we define the syntax of the logic pL_μ and the class of models given by Segala systems; in section 3 we define the denotational semantics of pL_μ as in [12, 2]; in section 4 we define the class of parity games that are going to be used to give game semantics to the logic; in section 5 we define the game semantics of pL_μ in terms of two player stochastic parity games; in section 6 we state the main theorem which asserts the equivalence of the denotational and game semantics. An extended version of this paper with detailed proofs is available at [11].

2 The Probabilistic Modal μ-Calculus

Given a set *Var* of propositional variables ranged over by the letters X, Y, Z and a set of labels L ranged over by the letters a, b, c, the formulae of the logic are defined by the following grammar:

$$F, G ::= X \mid \langle a \rangle F \mid [a]F \mid F \vee G \mid F \wedge G \mid \mu X.F \mid \nu X.F$$

We assume the usual notions of free and bound variables. A formula is closed if it has no free variables.

Definition 2.1 (Subformulae)**.** We define the function $Sub(F)$ by case analysis on F as follows:

$$Sub(X) \stackrel{\text{def}}{=} \{X\} \qquad Sub(F_1 \wedge F_2) \stackrel{\text{def}}{=} \{F_1 \wedge F_2\} \cup Sub(F_1) \cup Sub(F_2)$$
$$Sub([a]F) \stackrel{\text{def}}{=} \{[a]F\} \cup Sub(F) \qquad Sub(\nu X.F) \stackrel{\text{def}}{=} \{\nu X.F\} \cup Sub(F)$$

The cases for the connectives $\vee$, $\langle a \rangle$ and μX are defined as for their duals. We say that G is a subformula of F if $G \in Sub(F)$.
Definition 2.2 (Normal Formula)**.** A formula F is normal if

- Whenever $\star_1 X_1$ and $\star_2 X_2$, with $\star_1, \star_2 \in \{\mu, \nu\}$, are two different occurrences of binders in F then $X_1 \neq X_2$.
- No occurrence of a free variable X is also used in a binder $\star X$ in F.

Every formula can be put in normal form by standard α-renaming of the bound variables. We only consider formulae F in normal form. A bound variable X in F is called a μ-variable (respectively a

ν-variable) if it is bound in F by a μ (respectively ν) operator.

Definition 2.3 (Variables subsumption)**.** Given a normal formula F such that $\star_1 X_1.F_1, \star_2 X_2.F_2 \in Sub(F)$, we say that the variable X_1 subsumes X_2 in F if $\star_2 X_2.F_2 \in Sub(F_1)$.

The formulae of the logic pL_μ are interpreted over the class of models given by Segala systems.

Definition 2.4. A Segala system is a pair $\langle P, \{\xrightarrow{a}\}_{a\in L}\rangle$ where P is a *countable* set of states and for each $a \in L$, $\xrightarrow{a} \subseteq P \times \mathscr{D}(P)$ is the transition relation where $\mathscr{D}(P)$ is the set of all probability distribution over P. Given a probability distribution $d \in \mathscr{D}(P)$ we denote by $supp(d)$ the *support* of d defined as the set $\{p \in P \mid d(p) > 0\}$.

3 Denotational Semantics

Given a Segala system $\langle P, \{\xrightarrow{a}\}_{a\in L}\rangle$ we denote by $(P \to [0,1])$ and by $(\mathscr{D}(P) \to [0,1])$ the complete lattice of functions from P and from $\mathscr{D}(P)$ respectively, to the real interval $[0,1]$ with the component-wise order. Given a function $f \in (P \to [0,1])$, we denote by $\overline{f} \in (\mathscr{D}(P) \to [0,1])$ the lifted function defined as follows: $\overline{f} \stackrel{\text{def}}{=} \lambda d.\left(\sum_{p\in supp(d)} d(p)\cdot f(p)\right)$.
A function $\rho : Var \to (P \to [0,1])$ is called an *interpretation* of the variables. Given a function $f \in (P \to [0,1])$ we denote by $\rho[f/X]$ the interpretation that assigns f to the variable X, and $\rho(Y)$ to all other variables Y.
Fix a Segala System $\langle P, \{\xrightarrow{a}\}_{a\in L}\rangle$ and an interpretation ρ, the denotational semantics $[\![F]\!]_\rho : P \to [0,1]$ of the pL_μ formula F, under the interpretation ρ, is defined by structural induction on F as follows:

$$[\![X]\!]_\rho = \rho(X)$$

$$[\![G \vee H]\!]_\rho = [\![G]\!]_\rho \sqcup [\![H]\!]_\rho \quad [\![G \wedge H]\!]_\rho = [\![G]\!]_\rho \sqcap [\![H]\!]_\rho$$

$$[\![\langle a\rangle G]\!]_\rho = \lambda p.\Big(\bigsqcup\{\,\overline{[\![G]\!]_\rho}(d) \mid p \xrightarrow{a} d\}\Big) \quad [\![[a]H]\!]_\rho = \lambda p.\Big(\bigsqcap\{\,\overline{[\![H]\!]_\rho}(d) \mid p \xrightarrow{a} d\}\Big)$$

$$[\![\mu X.G]\!]_\rho = \text{lfp of the functional } \lambda f.([\![G]\!]_{\rho[f/X]}) \quad [\![\nu X.H]\!]_\rho = \text{gfp of the functional } \lambda f.([\![H]\!]_{\rho[f/X]})$$

Since the interpretation assigned to every pL_μ operator is monotone, the existence of the least and greatest fixed points is guaranteed by the Knaster-Tarski theorem. Moreover the least and the greatest fixed point can be computed inductively: $[\![\mu X.G]\!]_\rho = \bigsqcup_\alpha [\![\mu X.G]\!]^\alpha_\rho$ and $[\![\nu X.G]\!]_\rho = \bigsqcap_\alpha [\![\nu X.G]\!]^\alpha_\rho$ where

$$[\![\mu X.G]\!]^\alpha_\rho \stackrel{\text{def}}{=} \begin{cases} [\![G]\!]_{\rho[\lambda p.0/X]} & \text{for } \alpha = 0 \\ [\![G]\!]_{\rho[[\![\mu X.G]\!]^{\alpha-1}_\rho/X]} & \text{for } \alpha \text{ successor ordinal.} \\ [\![G]\!]_{\rho[\bigsqcup_{\beta<\alpha}[\![\mu X.G]\!]^\beta_\rho/X]} & \text{for } \alpha \text{ limit ordinal.} \end{cases} \qquad [\![\nu X.G]\!]^\alpha_\rho \stackrel{\text{def}}{=} \begin{cases} [\![G]\!]_{\rho[\lambda p.1/X]} & \text{for } \alpha = 0 \\ [\![G]\!]_{\rho[[\![\nu X.G]\!]^{\alpha-1}_\rho/X]} & \text{for } \alpha \text{ successor ordinal.} \\ [\![G]\!]_{\rho[\bigsqcap_{\beta<\alpha}[\![\nu X.G]\!]^\beta_\rho/X]} & \text{for } \alpha \text{ limit ordinal.} \end{cases}$$

4 Two Player Stochastic Parity Games

A turn-based Stochastic Game Arena (or just a $2\frac{1}{2}$ Game Arena) is a tuple $A = \langle (S,E), \{S_1, S_2, S_P\}, \pi\rangle$ where (S,E) is a directed graph with *countable* set of states S and successor function $E : S \to 2^S$; the sets S_1, S_2, S_P are a partition of S and $\pi : S_P \to \mathscr{D}(S)$ is called the probabilistic transition function. For every state $s \in S$, $E(s)$ is the (possibly infinite) set of successors of s. We require that for all $s \in S_P$, $E(s) = supp(\pi(s)) \neq \emptyset$. We denote by S_t the set of *terminal* states, i.e. those $s \in S$ such that $E(s) = \emptyset$.

The states in S_1 are Player 1 states; the states in S_2 are Player 2 states; the states in S_P are probabilistic, or Player P, states. At a state $s \in S_1$ (respectively $s \in S_2$), if $s \notin S_t$ Player 1 (respectively Player 2) chooses a successor from the set $E(s)$; if $s \in S_t$ the game ends. At a state $s \in S_P$, a successor state is chosen probabilistically according to the distribution $\pi(s)$.

A finite path $\vec{s}$ in A is a finite sequence $s_0,...,s_n$ of states in S such that for every $0 < i \leq n$, $s_i \in E(s_{i-1})$. An infinite path $\vec{s}$ in A is an infinite sequence of states $\{s_i\}_{i\in\mathbb{N}}$ such that for every $i > 0$, $s_i \in E(s_{i-1})$. We denote by $\mathscr{P}^\omega$ and $\mathscr{P}^{<\omega}$ the sets of infinite paths and finite paths in A respectively. Given a finite path $\vec{s} \in \mathscr{P}^{<\omega}$ we denote by $last(\vec{s})$ the last state $s \in S$ of $\vec{s}$. We denote by $\mathscr{P}_1^{<\omega}$, $\mathscr{P}_2^{<\omega}$ and $\mathscr{P}_P^{<\omega}$ the sets of finite paths having last state in S_1, S_2 and S_P respectively. We also denote by $\mathscr{P}^t$ the set of finite paths ending in a terminal state, i.e. the set of paths $\vec{s}$ such that $E(last(\vec{s})) = \emptyset$; the paths in $\mathscr{P}^t$ are called *terminated* paths. We denote by $\mathscr{P}$ the set $\mathscr{P}^\omega \cup \mathscr{P}^t$ and we refer to this set as the set of the possible *Plays* in A. Given a finite path $\vec{s} \in \mathscr{P}^{<\omega}$, we denote by $O_{\vec{s}}$ the set of all plays having $\vec{s}$ as prefix. We consider the standard topology on $\mathscr{P}$, where the basis for the open sets is given by the *cones* in G, i.e. the sets $O_{\vec{s}}$ for $\vec{s} \in \mathscr{P}^{<\omega}$.

As usual in Game Theory, Players' moves are determined by strategies. A *(full memory) deterministic* strategy σ_1 for Player 1 in the Game Arena A is defined as usual as a function $\sigma_1 : \mathscr{P}_1^{<\omega} \to S \cup \{\bullet\}$ such that $\sigma_1(\vec{s}) \in E(last(s))$ if $E(last(\vec{s})) \neq \emptyset$ and $\sigma_1(\vec{s}) = \bullet$ otherwise. Similarly a strategy σ_2 for Player 2 is defined as a function $\sigma_2 : \mathscr{P}_2^{<\omega} \to S \cup \{\bullet\}$.

A pair $\langle \sigma_1, \sigma_2 \rangle$ of strategies, one for each player, is called a strategy profile and determines the behaviors of both players. Fix a strategy profile $\langle \sigma_1, \sigma_2 \rangle$ and an initial state $s \in S$ we denote by $M^s_{\sigma_1,\sigma_2}$ the Markov Chain obtained by pruning A, starting from s, accordingly with σ_1 and σ_2. We often refer to this Markov Chain as the *Markov Play* generated by the strategy profile $\langle \sigma_1, \sigma_2 \rangle$ from $s \in S$. Each Markov Play $M^s_{\sigma_1,\sigma_2}$ has an associated probability measure on the set $\mathscr{P}$ of plays in A, which we denote by $\mathscr{M}^s_{\sigma_1,\sigma_2}$. The probability measure $\mathscr{M}^s_{\sigma_1,\sigma_2}$ is defined as usual as the *unique* probability measure which assigns to every basic open set $O_{\vec{s}}$ the multiplication of all the probabilities associated with the probabilistic transitions in $\vec{s}$.

A *priority assignment* $\mathbb{P}$ (of rank $n \in \mathbb{N}$) for the arena A is a function assigning a natural number in $\{0,...,n\}$ to every state in S, i.e. $\mathbb{P} : S \to \{0,...,n\}$. Given a priority assignment $\mathbb{P}$ of any rank, and an infinite path $\vec{s} = \{s_i\}_{i\in\mathbb{N}}$, we denote by $\mathbb{P}(\vec{s})$ the greatest natural number appearing infinitely often in the infinite sequence $\{\mathbb{P}(s_i)\}_{i\in\mathbb{N}}$. A *reward assignment* $\mathbb{B}$ for the arena A is a function assigning a value in the real interval $[0,1]$ to each terminal state $s \in S_t$, i.e. $\mathbb{B} : S_t \to [0,1]$. A pair $\langle \mathbb{P}, \mathbb{B} \rangle$ of a priority assignment $\mathbb{P}$ and a reward assignment $\mathbb{B}$ for the arena A, determines a unique measurable function $\Phi_{\langle \mathbb{P},\mathbb{B} \rangle} : \mathscr{P} \to [0,1]$ defined as follows:

$$\Phi_{\langle \mathbb{P},\mathbb{B} \rangle}(\vec{s}) = \begin{cases} \mathbb{B}(last(\vec{s})) & \text{if } \vec{s} \in \mathscr{P}^t \\ 0 & \text{if } \vec{s} \in \mathscr{P}^\omega \text{ and } \mathbb{P}(\vec{s}) \text{ is even} \\ 1 & \text{if } \vec{s} \in \mathscr{P}^\omega \text{ and } \mathbb{P}(\vec{s}) \text{ is odd} \end{cases}$$

The value $\Phi_{\langle \mathbb{P},\mathbb{B} \rangle}(\vec{s})$ should be understood as the payoff assigned to Player 1 when $\vec{s}$ is the outcome of the game. The payoff function $\Phi_{\langle \mathbb{P},\mathbb{B} \rangle}$ is called the *Parity payoff* determined by $\langle \mathbb{P}, \mathbb{B} \rangle$. We say that $\Phi_{\langle \mathbb{P},\mathbb{B} \rangle}$ has rank n if $\mathbb{P}$ has rank n. We often omit the subscript $\langle \mathbb{P}, \mathbb{B} \rangle$ in $\Phi_{\langle \mathbb{P},\mathbb{B} \rangle}$ if the context is clear enough. Note that, since the priority assigned by $\mathbb{P}$ to the terminal states $s \in S_t$ does not affect the induced payoff function, we can assume, without any loss of generality, that $\mathbb{P}(s) = 0$ for any terminal state $s \in S_t$. From now on we assume that this condition holds for every priority assignment $\mathbb{P}$.

A Two Player Stochastic Parity Game $\mathscr{G}$ is a tuple $\mathscr{G} = \langle A, \mathbb{P}, \mathbb{B} \rangle$ of a $2\frac{1}{2}$ Game Arena A, a priority assignment $\mathbb{P}$ and a $\mathbb{B}$ assignment for the arena A. Given a Parity game $\mathscr{G} = \langle A, \mathbb{P}, \mathbb{B} \rangle$, a state $s \in S$ and a

strategy profile $\langle \sigma_1, \sigma_2 \rangle$, we denote by with $\mathscr{M}^s_{\sigma_1,\sigma_2}(\Phi_{\langle \mathbb{P},\mathbb{B}\rangle})$ the value defined as:

$$\int_{\mathscr{P}} \Phi_{\langle \mathbb{P},\mathbb{B}\rangle}\, d\mathscr{M}^s_{\sigma_1,\sigma_2}$$

which corresponds to the expected payoff for Player 1 when the game starts in the state s and Player 1 and Player 2 follow the strategies σ_1 and σ_2 respectively. We denote by $Val_1(\mathscr{G}) : S \to [0,1]$ and $Val_2(\mathscr{G}) : S \to [0,1]$ the functions defined as follows:

$$Val_1(\mathscr{G})(s) = \sqcup_{\sigma_1} \sqcap_{\sigma_2} \mathscr{M}^s_{\sigma_1,\sigma_2}(\Phi) \qquad Val_2(\mathscr{G})(s) = \sqcap_{\sigma_2} \sqcup_{\sigma_1} \mathscr{M}^s_{\sigma_1,\sigma_2}(\Phi).$$

$Val_1(\mathscr{G})(s)$ represents the limit (expected) payoff that Player 1 can get, when the game begins in s, by choosing his strategy σ_1 first and then letting Player 2 pick an appropriate counter strategy σ_2. Similarly $Val_2(\mathscr{G})(s)$ represents the limit (expected) payoff that Player 1 can get, when the game begins in s, by first letting Player 2 choose a strategy σ_2 and then picking an appropriate counter strategy σ_1. Clearly $Val_1(\mathscr{G})(s) \leq Val_2(\mathscr{G})(s)$ for every $s \in S$.

Theorem 4.1 (Determinacy [8])**.** *For every Two Player Stochastic Parity Game $\mathscr{G} = \langle A, \mathbb{P}, \mathbb{B}\rangle$ the following equality holds:*

$$\forall s \in S,\ Val_1(\mathscr{G})(s) = Val_2(\mathscr{G})(s).$$

Intuitively the determinacy Theorem states that the players do not get any advantage by letting the opponent choose his strategy first. We just write $\mathbb{V}(\mathscr{G})$ for the *value* function of the Game defined as $Val_1(\mathscr{G}) = Val_2(\mathscr{G})$. As a corollary of the Determinacy theorem we have the following Lemma:

Lemma 4.2 (ε-optimal strategies)**.** *Given a Two Player Stochastic Parity Game $\mathscr{G} = \langle A, \mathbb{P}, \mathbb{B}\rangle$, for every $\varepsilon > 0$ the following assertions hold:*

- *there exists a strategy σ_1^ε for Player 1 such that for every $s \in S$, $\sqcap_{\sigma_2} \mathscr{M}^s_{\sigma_1^\varepsilon,\sigma_2}(\Phi) > \mathbb{V}(\mathscr{G})(s) - \varepsilon$.*
- *there exists a strategy σ_2^ε for Player 2 such that for every $s \in S$, $\sqcup_{\sigma_1} \mathscr{M}^s_{\sigma_1,\sigma_2^\varepsilon}(\Phi) < \mathbb{V}(\mathscr{G})(s) + \varepsilon$.*

5 Game Semantics

Fix a Segala System $\langle P, \{\xrightarrow{a}\}_{a\in L}\rangle$, a formula F and an interpretation $\rho : Var \to (P \to [0,1])$ of the variables, we denote by $\mathscr{G}^F_\rho$ the parity game $\langle A, \mathbb{P}, \mathbb{B}\rangle$ formally defined as described below.

The state space of the arena $A = \langle (S,E), \{S_1, S_2, S_P\}, \pi\rangle$, is the set $S = \big(P \cup \mathscr{D}(P)\big) \times Sub(F)$ of pairs of states $p \in P$ or distributions $d \in \mathscr{D}(P)$ and subformulae $G \in Sub(F)$; the transition relation E is defined as $E(\langle d, G\rangle) = \{\langle p, G\rangle \mid p \in supp(d)\}$ for every $d \in \mathscr{D}(P)$; $E(\langle p, G\rangle)$ is defined by case analysis on the outermost connective of G as follows:

1. if $G = X$, with X free in F, then $E(\langle p, G\rangle) = \emptyset$.
2. if $G = X$, with X bound in F by the subformula $\star X.H$, with $\star \in \{\mu, \nu\}$, then $E(\langle p, G\rangle) = \{\langle p, \star X.H\rangle\}$.
3. if $G = \star X.H$, with $\star \in \{\mu, \nu\}$, then $E(\langle p, G\rangle) = \{\langle p, H\rangle\}$.
4. if $G = \langle a\rangle H$ or $G = [a]H$ then $E(\langle p, G\rangle) = \{\langle d, H\rangle \mid p \xrightarrow{a} d\}$.
5. if $G = H \vee H'$ or $G = H \wedge H'$ then $E(\langle p, G\rangle) = \{\langle p, H\rangle, \langle p, H'\rangle\}$

The partition $\{S_1, S_2, S_P\}$ is defined as follows: every state $\langle p, G\rangle$ with G's main connective in $\{\langle a\rangle, \vee, \mu X\}$ or with $G = X$ where X is a μ-variable, is in S_1; dually every state $\langle p, G\rangle$ with G's main

connective in $\{[a], \wedge, \nu X\}$ or with $G = X$ where X is a ν-variable, is in S_2. Finally every state $\langle d, G\rangle$ is in S_P. The terminal states $\langle p, X\rangle$, with X free in F, are in S_1 by convention. The probability transition function $\pi : S_P \to S$ is defined as $\pi(\langle d, G\rangle)(\langle p, G\rangle) = d(p)$. The priority assignment $\mathbb{P}$ is defined as usual in μ-calculus model checking games [14]: the priority assigned to the states $\langle p, X\rangle$, with X a μ-variable is even; dually the priority assigned to the states $\langle p, X\rangle$, with X a ν-variable is odd. Moreover $\mathbb{P}(\langle p, X\rangle) > \mathbb{P}(\langle p', X'\rangle)$ if X subsumes X' in F. All other states get priority 0. The reward assigment $\mathbb{B}$ is defined as $\mathbb{B}(\langle p, X\rangle) = \rho(X)(p)$ for every terminal state $\langle p, X\rangle$ with X free in F. All other terminal states in $\mathscr{G}_\rho^F$ are either of the form $\langle p, \langle a\rangle H\rangle$ or $\langle p, [a] H\rangle$. The reward assignment $\mathbb{B}$ is defined on these terminal states as follows: $\mathbb{B}(\langle p, \langle a\rangle H\rangle) = 0$ and $\mathbb{B}(\langle p, [a] H\rangle) = 1$.

Fix a Segala system $\langle P, \{\xrightarrow{a}\}_{a \in L}\rangle$, the game semantics of the formula F under the interpretation ρ, is the function $(\!| F |\!)_\rho : P \to [0, 1]$ defined as $(\!| F |\!)_\rho \stackrel{\text{def}}{=} \lambda p. \mathbb{V}(\mathscr{G}_\rho^F)(\langle p, F\rangle)$.

6 Equivalence of Denotational and Game Semantics for pL_μ

We are now ready to give the main theorem which states that given any Segala system $\langle P, \{\xrightarrow{a}\}_{a \in L}\rangle$, the denotational and game semantics of any pL_μ formula coincide.

Theorem 6.1. *Given a Segala system $\langle P, \{\xrightarrow{a}\}_{a \in L}\rangle$, for every pL_μ formula F and interpretation ρ for the variables, the following equality holds: $[\![F]\!]_\rho = (\!| F |\!)_\rho$.*

References

[1] M. Dam. CTL* and ECTL* as fragments of the modal μ-calculus. In *Theoretical Computer Science, Volume 126, Issue 1*, pages 77,96. Elsevier Science B.V., 1994.

[2] L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences, Volume 68 , Issue 2*, pages 374 – 397, 2004.

[3] E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *SFCS '91: Proceedings of the 32nd annual symposium on Foundations of computer science*, pages 368–377. IEEE Computer Society, 1991.

[4] D. Fischer, E. Gradel, and L. Kaiser. Model checking games for the quantitative μ-calculus. In *Theory of Computing Systems*. Springer New York, 2009.

[5] Michael Huth and Marta Kwiatkowska. Quantitative analysis and model checking. In *LICS 1997*, page 111, Washington, DC, USA, 1997. IEEE Computer Society.

[6] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. *Lecture Notes in Computer Science*, 1119:263–277, 1996.

[7] D. Kozen. Results on the propositional mu-calculus. In *Theoretical Computer Science*, pages 333–354, 1983.

[8] D. A. Martin. The determinacy of Blackwell games. In *Journal of Symbolic Logic Volume 63, Issue 4, 1565-1581*, 1998.

[9] A. McIver and C. Morgan. Results on the quantitative mu-calculus qmu. *CoRR*, cs.LO/0309024, 2003.

[10] A. McIver and C. Morgan. Results on the quantitative μ-calculus qmμ. *ACM Trans. Comput. Logic*, 8(1):3, 2007.

[11] M. Mio. Extended version available at: http://homepages.inf.ed.ac.uk/s0792227/FICS2010extended.pdf.

[12] C. Morgan and A. McIver. A probabilistic temporal calculus based on expectations. In *In Lindsay Groves and Steve Reeves, editors, Proc. Formal Methods*. Springer Verlag, 1997.

[13] R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology, 1995.

[14] Colin Stirling. *Modal and temporal logics for processes*. Springer (Texts in Computer Science), 2001.

[15] Thomas Studer. On the proof theory of the modal mu-calculus. In *Studia Logica, Volume 89, Number 3*. Springer Netherlands, 2007.

Denotational semantics for lazy initialization of letrec

black holes as exceptions rather than divergence

Keiko Nakata
Institute of Cybernetics at Tallinn University of Technology

Abstract

We present a denotational semantics for a simply typed call-by-need letrec calculus, which distinguishes direct cycles, such as *let rec* $x = x$ *in* x and *let rec* $x = y$ *and* $y = x+1$ *in* x, and looping recursion, such as *let rec* $f = \lambda x.f\ x$ *in* $f\ 0$. In this semantics the former denote an exception whereas the latter denotes divergence.

The distinction is motivated by "lazy evaluation" as implemented in OCaml via *lazy*/*force* and Racket (formerly PLT Scheme) via *delay*/*force*: when a delayed variable is dereferenced for the first time, it is first pre-initialized to an exception-raising thunk and is updated afterward by the value obtained by evaluating the expression bound to the variable. Any attempt to dereference the variable during the initialization raises an exception rather than diverges. This way, lazy evaluation provides a useful measure to initialize recursive bindings by exploring a successful initialization order of the bindings at runtime and by signaling an exception when there is no such order. It is also used for the initialization semantics of the object system in the F# programming language.

The denotational semantics is proved adequate with respect to a referential operational semantics.

1 Introduction

Lazy evaluation is a well-known technique in practice to initialize recursive bindings. OCaml [6] and Racket (formerly PLT Scheme) [4], provide language constructs, *lazy*/*force* and *delay*/*force* operators respectively, to support lazy evaluation atop call-by-value languages with arbitrary side-effects. Their implementations are quite simple: when a delayed variable is dereferenced for the first time, it is first pre-initialized to an exception-raising thunk and is updated afterward by the value obtained by evaluating the expression bound to the variable. Any attempt to dereference the variable during the initialization raises an exception rather than diverges. In other words, lazy evaluation as implemented in OCaml and Racket distinguishes direct cycles [1], which we call "black holes", such as *let rec* $x = x$ *in* x and *let rec* $x = y$ *and* $y = x+1$ *in* x, and looping recursion, such as *let rec* $f = \lambda x.f\ x$ *in* $f\ 0$. The former raise an exception, whereas the latter diverges.

Lazy evaluation provides a useful measure to initialize recursive bindings by exploring a successful initialization order of the bindings at runtime and by signaling an exception when there is no such order. In [12], Syme advocates the use of lazy evaluation for initializing mutually recursive bindings in ML-like languages to permit a wider range of recursive bindings [2]. Flexibility in handling recursive bindings is particularly important for these languages to interface with external abstract libraries such as GUI APIs. Syme's proposal can be implemented using OCaml's *lazy*/*force* operators and it underlies the initialization semantics of the object system in F# [13].

There is a gap between lazy evaluation, as outlined above, and conventional models for lazy, or call-by-need, computation as found in the literature. Traditionally call-by-need is understood as an economical implementation of call-by-name, which does not distinguish black holes and looping recursion but typically interprets both uniformly as "undefined". The gap becomes evident when a programming language supports exception handling, as both OCaml and Racket do — one can catch exceptions but

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 61-67

[1] Direct cycles are also known as provable divergence.

[2] In ML, the right-hand side of recursive bindings is restricted to be syntactic values.

$$
\begin{array}{lrcl}
\textit{Expressions} & M,N & ::= & n \mid x \mid \lambda x.M \mid M\ N \mid \mathsf{let\ rec}\ x_1\ \mathsf{be}\ M_1,\ldots,x_n\ \mathsf{be}\ M_n\ \mathsf{in}\ M \mid \bullet \\
\textit{Results} & V & ::= & n \mid \lambda x.M \mid \bullet \\
\textit{Types} & \tau & ::= & \mathsf{nat} \mid \tau_1 \to \tau_2
\end{array}
$$

Figure 1: Syntax of λ_{letrec}

$$
n : \mathsf{nat} \qquad x : \mathsf{type}(x) \qquad \bullet : \tau
$$

$$
\frac{x : \tau_1 \quad M : \tau_2}{\lambda x.M : \tau_1 \to \tau_2} \qquad \frac{M : \tau_1 \to \tau_2 \quad N : \tau_1}{M\ N : \tau_2} \qquad \frac{x_1 : \tau_1 \quad \ldots \quad x_n : \tau_n \quad M_1 : \tau_1 \quad \ldots \quad M_n : \tau_n \quad N : \tau}{\mathsf{let\ rec}\ x_1\ \mathsf{be}\ M_1,\ldots,x_n\ \mathsf{be}\ M_n\ \mathsf{in}\ N : \tau}
$$

Figure 2: Typing rules

cannot catch divergence. Indeed catching exceptions due to black holes is perfectly acceptable, or could be even desired, in practice; it is just like catching null-pointer exceptions due to object initialization failure in object-oriented languages.

In this paper we present a denotational semantics, which matches the lazy evaluation as implemented in OCaml and Racket and used in F#'s object initialization. In this semantics, direct cycles denote exceptions whereas looping recursion denotes divergence. The key observation is to think of lazy evaluation as a most successful initialization strategy of recursive bindings: the initialization succeeds if and only if there is a non-circular order in which the bindings can be initialized. The operational semantics searches such an order by on-demand computation. The denotational semantics searches such one intuitively by initializing recursive bindings in parallel and choosing the most successful result as the denotation.

The denotational semantics, proved adequate with respect to a referential operational semantics, is the main contribution of the paper.

2 Syntax and operational semantics

The syntax of our simply typed letrec calculus, λ_{letrec}, is given in figure 1. An expression is either a natural number $n \in N$, variable x, abstraction $\lambda x.M$, application $M\ N$, letrec $\mathsf{let\ rec}\ x_1\ \mathsf{be}\ M_1,\ldots,x_n\ \mathsf{be}\ M_n\ \mathsf{in}\ M$, or black hole $\bullet$, which represents an exception. Results are natural numbers, abstraction and black holes. A type is either a base type, nat, or a function type of shape $\tau_1 \to \tau_2$. To simplify the calculus, we assume each variable x is associated with a unique type, given, e.g., $\mathsf{type}(x)$. Typing rules are found in figure 2, which are all straightforward.

In figure 3, we present the natural semantics. The natural semantics is identical to that given in our previous work [8], which is very much inspired by Launchbury's [5] and Sestoft's [10]. Heaps, ranged over by metavariables Ψ and Φ, are finite mappings from variables to expressions. We write $x_1 \mapsto M_1,\ldots,x_n \mapsto M_n$ to denote a heap whose domain is $\{x_1,\ldots,x_n\}$, and which maps x_i's to M_i's. The notation $\Psi[x_1 \mapsto M_1,\ldots,x_n \mapsto M_n]$ denotes mapping extension. Precisely, $\Psi[x_1 \mapsto M_1,\ldots x_n \mapsto M_n](x_i) = M_i$ and $\Psi[x_1 \mapsto M_1,\ldots x_n \mapsto M_n](y) = \Psi(y)$ when $y \neq x_i$ for any i in $1,\ldots,n$. We write $\Psi[x \mapsto M]$ to denote a single extension of Ψ with M at x. In rule *Letrec*, M_i''s and N' denote expressions obtained from M_i's and N by substituting x_i''s for x_i's, respectively. We may abbreviate $\langle\Psi\rangle M$ where Ψ is an empty mapping, i.e., the domain of Ψ is empty, to $\langle\rangle M$.

The judgment $\langle\Psi\rangle M \Downarrow \langle\Phi\rangle V$ expresses that an expression M in an initial heap Ψ evaluates to a result V with the heap being Φ. In *Variable* rule, the heap Ψ is updated to map x to $\bullet$ while the expression bound to x is evaluated. For instance, $\langle\rangle\ \mathsf{let\ rec}\ x\,\mathsf{be}\,x\ \mathsf{in}\ x \Downarrow \langle x' \mapsto \bullet\rangle \bullet$ is deduced. This way, an attempt to dereference a variable which is under "initialization" results in a black hole. $Error_\beta$ rule propagates black holes. Other rules are self-explanatory.

$$\textit{Result}\quad \langle\Psi\rangle V \Downarrow \langle\Psi\rangle V$$

$$\textit{Application}\quad \frac{\langle\Psi\rangle M_1 \Downarrow \langle\Phi\rangle \lambda x.N \quad \langle\Phi[x' \mapsto M_2]\rangle N[x'/x] \Downarrow \langle\Psi'\rangle V \quad x' \text{ fresh}}{\langle\Psi\rangle M_1\ M_2 \Downarrow \langle\Psi'\rangle V}$$

$$\textit{Variable}\quad \frac{\langle\Psi[x \mapsto \bullet]\rangle \Psi(x) \Downarrow \langle\Phi\rangle V}{\langle\Psi\rangle x \Downarrow \langle\Phi[x \mapsto V]\rangle V}$$

$$\textit{Letrec}\quad \frac{\langle\Psi[x'_1 \mapsto M'_1, \ldots, x'_n \mapsto M'_n]\rangle N' \Downarrow \langle\Phi\rangle V \quad x'_1, \ldots, x'_n \text{ fresh}}{\langle\Psi\rangle \mathsf{let\ rec}\ x_1\,\mathsf{be}\,M_1, \ldots, x_n\,\mathsf{be}\,M_n\ \mathsf{in}\ N \Downarrow \langle\Phi\rangle V}$$

$$\textit{Error}_\beta\quad \frac{\langle\Psi\rangle M_1 \Downarrow \langle\Phi\rangle \bullet}{\langle\Psi\rangle M_1\ M_2 \Downarrow \langle\Phi\rangle \bullet}$$

Figure 3: Natural semantics

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle x' \mapsto \bullet, f' \mapsto \bullet\rangle \lambda y.y \Downarrow \langle x' \mapsto \bullet, f' \mapsto \bullet\rangle \lambda y.y} \quad \dfrac{}{\langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}}{\langle x' \mapsto \bullet, f' \mapsto \lambda y.y\rangle f' \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y\rangle \lambda y.y} \quad \dfrac{}{\langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle x' \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}}{\langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto x'\rangle y' \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}}{\langle x' \mapsto \bullet, f' \mapsto \lambda y.y\rangle f'\ x' \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}}{\langle x' \mapsto f'\ x', f' \mapsto \lambda y.y\rangle x' \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}}{\langle\rangle\ \mathsf{let\ rec}\ x\,\mathsf{be}\,f\ x, f\,\mathsf{be}\,\lambda y.y\ \mathsf{in}\ x \Downarrow \langle x' \mapsto \bullet, f' \mapsto \lambda y.y, y' \mapsto \bullet\rangle \bullet}$$

Figure 4: The derivation for $\mathsf{let\ rec}\ x\,\mathsf{be}\,f\ x, f\,\mathsf{be}\,\lambda y.y\ \mathsf{in}\ x$

In figure 4 we present the derivation for the expression $\mathsf{let\ rec}\ x\,\mathsf{be}\,f\ x, f\,\mathsf{be}\,\lambda y.y\ \mathsf{in}\ x$. We deliberately chose a black hole producing expression.

3 Denotational semantics

We proceed to the denotational semantics. An expression M of type τ denotes an element of $(V_\tau + \mathsf{Err}_\tau)_\perp$, where $(\cdot)_\perp$ is lifting and Err_τ is a singleton, whose only element is $\bullet_\tau$. V_τ denotes proper values of type τ and is defined by induction on τ:

$$V_{\mathsf{nat}} = N \qquad V_{\tau_0 \to \tau_1} = [(V_{\tau_0} + \mathsf{Err}_{\tau_0})_\perp \to (V_{\tau_1} + \mathsf{Err}_{\tau_1})_\perp]$$

We omit injections for both the lifting and the sum. A metavariable φ ranges over proper function values, i.e., elements of $V_{\tau_0 \to \tau_1}$ for some τ_0 and τ_1. For $d \in (V_{\tau_0 \to \tau_1} + \mathsf{Err}_{\tau_0 \to \tau_1})_\perp$ and $d' \in (V_{\tau_0} + \mathsf{Err}_{\tau_0})_\perp$, application of d to d' is defined by

$$d(d') = \begin{cases} \perp_{\tau_1} & \text{when } d = \perp_{\tau_0 \to \tau_1} \\ \bullet_{\tau_1} & \text{when } d = \bullet_{\tau_0 \to \tau_1} \\ \varphi(d') & \text{when } d = \varphi \end{cases}$$

$$
\begin{array}{rcl}
[\![n:\tau]\!]_\rho & = & n \\
[\![x:\tau]\!]_\rho & = & \rho(x) \\
[\![\bullet:\tau]\!]_\rho & = & \bullet_\tau \\
[\![\lambda x.M:\tau_0\to\tau_1]\!]_\rho & = & \lambda v.[\![M:\tau_1]\!]_{\rho[x\mapsto v]} \\
[\![M^{\tau_0\to\tau_1}\ N^{\tau_0}:\tau_1]\!]_\rho & = & ([\![M:\tau_0\to\tau_1]\!]_\rho)([\![N:\tau_0]\!]_\rho) \\
[\![\mathsf{let\ rec}\ x_1\ \mathsf{be}M_1^{\tau_1},\ldots,x_n\ \mathsf{be}M_n^{\tau_n}\ \mathsf{in}\ N:\tau]\!]_\rho & = & [\![N:\tau]\!]_{\{\{x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}\}\}_\rho^{(n)}} \\
\{\{x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}\}\}_\rho^{(m+1)} & = & \\
\multicolumn{3}{r}{\mu\rho'.\rho[x_1\mapsto [\![M_1:\tau_1]\!]_{\rho_m}\cdot[\![M_1:\tau_1]\!]_{\rho'},\ldots,x_n\mapsto [\![M_n:\tau_n]\!]_{\rho_m}\cdot[\![M_n:\tau_n]\!]_{\rho'}]} \\
\multicolumn{3}{r}{\textit{where}\ \rho_m=\{\{x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}\}\}_\rho^{(m)}} \\
\{\{x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}\}\}_\rho^{(0)} & = & \rho[x_1\mapsto\bullet_{\tau_1},\ldots,x_n\mapsto\bullet_{\tau_n}]
\end{array}
$$

Figure 5: Denotational semantics of λ_{letrec}

Moreover we write $(d)^*$ to denote the strict version of d on both $\bot_{\tau_0}$ and $\bullet_{\tau_0}$, i.e.,

$$
(d)^*(d') = \begin{cases} \bot_{\tau_1} & \text{when } d=\varphi \text{ and } d'=\bot_{\tau_0} \\ \bullet_{\tau_1} & \text{when } d=\varphi \text{ and } d'=\bullet_{\tau_0} \\ d(d') & \text{otherwise} \end{cases}
$$

An environment, ρ, is a function from variables to denotations, which respects types, i.e., $\rho(x)\in(V_\tau+\mathsf{Err}_\tau)_\bot$ where $x:\tau$. The least environment, $\rho_\bot$, maps all variables to bottom elements.

The semantic function $[\![M:\tau]\!]_\rho$ assigns a denotation to a typing derivation $M:\tau$ under an environment ρ and is defined in figure 5 by induction on the derivation [3]. μ stands for the least fixed point operator. $\rho[x_1\mapsto d_1,\ldots,x_n\mapsto d_n]$ stands for extension of ρ with d_i's at x_i's. For $d,d'\in(V_\tau+\mathsf{Err}_\tau)_\bot$, the notation $d\cdot d'$ abbreviates $((\lambda y.\lambda x.x)^*(d))(d')$. The semantic function, being defined using only continuous operations, is continuous and mostly standard (c.f., [14]) except for letrec. The denotation of $\mathsf{let\ rec}\ x_1\ \mathsf{be}M_1^{\tau_1},\ldots,x_n\ \mathsf{be}M_n^{\tau_n}\ \mathsf{in}\ N:\tau$ is defined with the help of a semantic function for (typed) heaps. The function $\{\{x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}\}\}_\rho^{(m)}$ takes three parameters, a heap $x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}$, an environment ρ and a natural number m, and returns an environment. It is defined by induction on m, with semantic functions for $M_i:\tau_i$'s given by the outer induction.

We compute the denotation of a heap $\Psi=x_1\mapsto M_1^{\tau_1},\ldots,x_n\mapsto M_n^{\tau_n}$ under an environment ρ as follows. Let $\rho_m=\{\{\Psi\}\}_\rho^{(m)}$. The variables x_i's are first pre-initialized to black holes, that is, $\rho_0=\rho[x_1\mapsto\bullet_{\tau_1},\ldots,x_n\mapsto\bullet_{\tau_n}]$. Next we compute the denotation $[\![M_i:\tau_i]\!]_{\rho_0}$ of $M_i:\tau_i$ for each i under the initial environment ρ_0, so that we take the fixed-point semantics for the recursive bindings whose initialization was successful. That is, $\rho_1=\mu\rho'.\rho[x_1\mapsto d_1,\ldots,x_n\mapsto d_n]$ where

$$
d_i = \begin{cases} \bullet_{\tau_i} & \textit{when}\ \ [\![M_i:\tau_i]\!]_{\rho_0}=\bullet_{\tau_i} \\ [\![M_i:\tau_i]\!]_{\rho'} & \textit{otherwise} \end{cases}
$$

Indeed it follows from lemmata 3.1 and 3.2 below that this is equivalent to defining $\rho_1=\mu\rho'.\rho[x_1\mapsto [\![M_1:\tau_1]\!]_{\rho_0}\cdot[\![M_1:\tau_1]\!]_{\rho'},\ldots,x_n\mapsto [\![M_n:\tau_n]\!]_{\rho_0}\cdot[\![M_n:\tau_n]\!]_{\rho'}]$. Generally, ρ_{m+1} is given by taking the fixed-point semantics for the recursive bindings whose initialization is successful under the environment ρ_m; i.e., $\rho_{m+1}=\mu\rho'.\rho[x_1\mapsto d_1,\ldots,x_n\mapsto d_n]$ where

$$
d_i = \begin{cases} \bullet_{\tau_i} & \textit{when}\ \ [\![M_i:\tau_i]\!]_{\rho_m}=\bullet_{\tau_i} \\ [\![M_i:\tau_i]\!]_{\rho'} & \textit{otherwise} \end{cases}
$$

[3] We use the lambda notation to express (mathematical) functions on domains.

This process is iterated for n times, where n is the length of the heap Ψ. The number of the iteration is justified, as it converges by then: $\{\{\Psi\}\}_\rho^{(n)} = \{\{\Psi\}\}_\rho^{(n+m)}$ for any m (lemma 3.3 below). Let us define $\{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho = \{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(n)}$. For instance, we have $\{\{x \mapsto x^{\mathsf{nat}}\}\}_{\rho_\perp}(x) = \bullet_{\mathsf{nat}}$, $\{\{f \mapsto (\lambda x.f\ x)^{\mathsf{nat}\to\mathsf{nat}}\}\}_{\rho_\perp}(f) = \lambda x.\perp_{\mathsf{nat}}$, and $\{\{x \mapsto (f\ x)^{\mathsf{nat}}, f \mapsto (\lambda y.y)^{\mathsf{nat}\to\mathsf{nat}}\}\}_{\rho_\perp} = \rho_\perp[x \mapsto \bullet_{\mathsf{nat}}, f \mapsto \lambda y.y]$ therefore $[\![\mathsf{let\ rec}\ x\,\mathsf{be}\,(f\ x)^{\mathsf{nat}}, f\,\mathsf{be}\,(\lambda y.y)^{\mathsf{nat}\to\mathsf{nat}}\ \mathsf{in}\ x : \mathsf{nat}]\!]_{\rho_\perp} = \bullet_{\mathsf{nat}}$.

3.1 Adequacy

The denotational semantics is adequate with respect to the natural semantics. An important fact, to be stated in lemma 3.4, is that, as we iteratively compute denotations of (typed) heaps Ψ under an environment ρ, we reach a fixed point: $\{\{\Psi\}\}_\rho(x) = [\![\Psi(x)]\!]_{\{\{\Psi\}\}_\rho}$.

We define a relation $\ll_\tau \subseteq (V_\tau + \mathsf{Err}_\tau)_\perp \times (V_\tau + \mathsf{Err}_\tau)_\perp$ by induction on τ:

$$\begin{array}{l}
\bullet_\tau \ll_\tau d \quad \text{for any } d \in (V_\tau + \mathsf{Err}_\tau)_\perp \\
\perp_\tau \ll_\tau \perp_\tau \\
n \ll_{\mathsf{nat}} n \\
\varphi \ll_{\tau_1\to\tau_2} \varphi' \ \text{ iff } \ d \ll_{\tau_1} d' \text{ implies } \varphi(d) \ll_{\tau_2} \varphi'(d')
\end{array}$$

Then a relation $\ll$ on environments is defined such that $\rho \ll \rho'$ iff for any x, $\rho(x) \ll_\tau \rho'(x)$ where $x : \tau$. It captures a relationship between (intermediate) denotations of a heap at different iterations (see lemma 3.2 below).

Lemma 3.1. *For any $\rho, \rho', M : \tau$, if $\rho \ll \rho'$, then $[\![M : \tau]\!]_\rho \ll_\tau [\![M : \tau]\!]_{\rho'}$*

Lemma 3.2. *For any $x_1 : \tau_1, \ldots, x_n : \tau_n, M_1 : \tau_1, \ldots, M_n : \tau_n, m$ and ρ, $\{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(m)} \ll \{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(m+1)}$.*

Lemma 3.3. *For any $x_1 : \tau_1, \ldots, x_n : \tau_n, M_1 : \tau_1, \ldots, M_n : \tau_n, m$ and ρ, $\{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(n)} = \{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(n+m)}$.*

Lemma 3.4. *For any $x_1 : \tau_1, \ldots, x_n : \tau_n, M_1 : \tau_1, \ldots, M_n : \tau_n, \rho$ and $i \in 1..n$, $\{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(n)}(x_i) = [\![M_i : \tau_i]\!]_{\{\{x_1 \mapsto M_1^{\tau_1}, \ldots, x_n \mapsto M_n^{\tau_n}\}\}_\rho^{(n)}}$.*

The natural semantics is correct with respect to the denotational semantics in that evaluations preserve the denotations of expressions.

Proposition 3.1. *For any typed expression $M : \tau$, if $\langle\rangle\, M \Downarrow \langle\Psi\rangle\, V$, then $V : \tau$ and $[\![M : \tau]\!]_{\rho_\perp} = [\![V : \tau]\!]_{\{\{\Psi\}\}_{\rho_\perp}}$.*

Moreover an expression evaluates to a result if and only if its denotation is non-bottom.

Proposition 3.2. *For any typed expression $M : \tau$, $[\![M : \tau]\!]_{\rho_\perp} \neq \perp_\tau$ iff there are Φ and V such that $\langle\rangle\, M \Downarrow \langle\Phi\rangle\, V$.*

4 Related work

The natural semantics used in the paper is very much inspired by those of Launchbury [5] and Sestoft [10]. Ariola and Felleisen gave a reduction semantics for λ_{letrec} [2], which is proved equivalent to our natural semantics [8]. Our denotational semantics is also influenced by Launchbury's, except that his

semantics assigns a bottom element to both black holes and looping recursion; the distinction of the two is at the heart of our work.

Ariola and Klop [3] studied equational theories of cyclic lambda calculi by means of cyclic lambda graphs and observed that having non-restricted substitution leads to non-confluence. Ariola and Blom [1] and Schmidt-Schauß et al. [9] use infinite lambda terms to reason about call-by-need letrec; it is not obvious if their techniques can be adapted so that black holes are distinguished from divergence.

Viewing black holes as exceptions is not new. For instance, Moggi and Sabry's monadic operational semantics for value recursion signals a monadic error when a black hole is encountered [7]. The idea seems to be ascribed to the backpatching semantics of Scheme [11].

5 Conclusion

We have presented a denotational semantics for lazy initialization of letrec. The semantics interprets direct cycles, called black holes, as exceptions, which fits lazy evaluation as implemented in OCaml and Racket and which underlies the initialization semantics of F#'s object system. We think signaling an exception for these "non-sense recursion" is natural and useful in practice; we believe it is also natural in theory.

Acknowledgments I thank Matthias Felleisen and Olivier Danvy for valuable exchanges, Eijiro Sumii, Makoto Tatsuta, Masahito Hasegawa, Yukiyoshi Kameyama and Yasuhiko Minamide for discussions.

This research was supported by the Estonian Centre of Excellence in Computer Science, EXCS, funded by the European Regional Development Fund, and the Estonian Science Foundation grant no. 6940.

References

[1] Z. Ariola and S. Blom. Cyclic lambda calculi. In M. Abadi and T. Ito, editors, *Proc. of 3rd Int. Symp. on Theoretical Aspects of Computer Software (TACS 1997)*, volume 1281 of *Lect. Notes in Comput. Sci.*, pages 77–106. Springer, 1997.

[2] Z. Ariola and M. Felleisen. The call-by-need lambda calculus. *J. Funct. Program.*, 7(3):265–301, 1997.

[3] Z. Ariola and J. Klop. Cyclic lambda graph rewriting. In *Proc. of 9th Symp. on Logic in Computer Science (LICS 1994)*, pages 416–425. IEEE Computer Society, 1994.

[4] Matthew Flatt and PLT. Reference: PLT Scheme. Technical Report PLT-TR2010-reference-v4.2.5, PLT Scheme Inc., April 2010. http://plt-scheme.org/techreports/ (PLT Scheme is now Racket.).

[5] J. Launchbury. A natural semantics for lazy evaluation. In *Conf. Record of 20th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL 1993)*, pages 144–154. ACM Press, 1993.

[6] X. Leroy, D. Doligez, J. Garrigue, D. Rémy, and J. Vouillon. The Objective Caml system, release 3.11, November 2008. http://caml.inria.fr/.

[7] E. Moggi and A. Sabry. An abstract monadic semantics for value recursion. *Informatique Théorique et Applications*, 38(4):375–400, 2004.

[8] K. Nakata and M. Hasegawa. Small-step and big-step semantics for call-by-need. *J. Funct. Program.*, 19(6):699–722, 2009.

[9] M. Schmidt-Schauß, D. Sabel, and E. Machkasova. Simulation in the call-by-need lambda-calculus with letrec. In C. Lynch, editor, *Proc. of 21st Int. Conf. on Rewriting Techniques and Applications (RTA 2010)*, volume 6 of *LIPIcs*, pages 295–310. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[10] P. Sestoft. Deriving a lazy abstract machine. *J. Funct. Program.*, 7(3):231–264, 1997.

[11] M. Sperber, K. Dybvig, M. Flatt, A. Straaten, R. Findler, and J. Matthews. Revised^6 report on the algorithmic language scheme. *J. Funct. Program.*, 19(S1):1–301, 2009.

[12] D. Syme. Initializing mutually referential abstract objects: The value recursion challenge. In N. Benton and X. Leory, editors, *Proc. of the ACM-SIGPLAN Workshop on ML (ML 2005)*, volume 148 of *Electr. Notes Theor. Comput. Sci.*, pages 3–25. Elsevier, 2006.

[13] D. Syme and the MSR F# team. The F# programming language, version 2.0.0.0, April 2010. http://research.microsoft.com/en-us/um/cambridge/projects/fsharp/.

[14] G. Winskel. *The Formal Semantics of Programming Languages: An Introduction*. The MIT Press, 1993.

Fixed Point Argument and Tilings without Long Range Order

Andrei Romashchenko
LIF de Marseille (CNRS & Univ. Aix–Marseille) on leave from IITP RAS of Moscow
andrei.romashchenko@lif.univ-mrs.fr

Abstract

Tiling is a classic model of combinatorial structure, whose global properties are determined by simple local rules. In this paper we construct tilings with some specific properties, using self-referential argument originating in Kleene's recursion theorem. This methods proves to be very effective though the initial problems under consideration are purely combinatorial and do not refer explicitly to any questions of computability.

More specifically, we investigate the problem of *long range order* formulated by C. Radin: whether one can determine by local rules such tilings on the plane in which we loose all information while traveling between distant areas. We present a partial answer to Radin's question by constructing a tile set $\tau = \tau_0 \sqcup \tau_1$ (i.e., all tiles in τ are divided into two disjoint classes of 0-tiles and 1-tiles) such that every τ-tiling of the plane corresponds to some configuration in $\{0,1\}^{\mathbb{Z}^2}$, where there is no long range order.

1 Introduction

In this paper we call by *tile* a unit square with colored sides. Given a finite set of tiles we consider tilings, i.e., coverings of the plane by copies of these tiles such that colors match (adjacent sides of every two neighbor tiles must have the same color in both squares).

For example, for a tile set that consists of two tiles (the first one has black lower and left side and white top and right sides, and the second one has the opposite colors) only the checkerboard tilings are possible.

More formally, we consider a finite set C of colors. A *tile* is a quadruple of colors (assigned to its four sides), i.e., an element of C^4. A *tile set* is a subset $\tau \subset C^4$. Respectively, a *tiling* of the plane with tiles from τ (τ-*tiling*) is a mapping $U\colon \mathbb{Z}^2 \to \tau$ that respects the color matching condition.

A tiling U is *periodic* if it has a *period*, i.e., a non-zero vector $T \in \mathbb{Z}^2$ such that $U(x+T) = U(x)$ for all $x \in \mathbb{Z}^2$. For example, for the checkerboard tiling in the example above the vertical and horizontal unit shifts are periods.

Intuitively it seems that a periodic tiling has an extremely simple structure. It is not surprising that some local rules enforce simple and regular periodic configurations. On the contrary, what is rather unexpected, is the fact that some tile sets do imply much more complex global structures. The first result of this kind was proven by Berger in [2]:

Theorem 1. *There exists a tile set τ such that τ-tilings exist and all of them are aperiodic* (*such tilings are called* aperiodic).

Berger's theorem shows that very simple local rules (matching rules for some finite set of colored tiles) can imply rather nontrivial global structure. Later it was shown that a tile set can enforce even much more sophisticated structures than just aperiodic configurations. For example, in [4] there was constructed a tile set whose tilings have maximal "density of information" (more precisely, Kolmogorov complexity of each $N \times N$-square in a tiling must have Kolmogorov complexity $\Omega(N)$).

Charles Radin in [11] noticed that there exist another natural angle of the intuitive idea of "regular configuration". A global configuration looks regular if it has *long range order*, i.e., there is dependency

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 69-75

even between very far remote tiles. For example, in a periodic tiling U every two tiles $U(x)$ and $U(x+T)$ must be equal to each other if T is a multiple of the minimal period (thus, T can be arbitrarily large).

Following [11], we define a notion of *long range order* in a more general way. Let τ be a tile sets and P be some subset of τ. For a τ-tiling U we denote by $\nu(P)$ the frequency of tiles from P in U. More precisely, for each positive integer N we count in U the fraction of tiles from P in N-neighborhood of the point $(0,0)$. We define $\nu(P)$ as the limit of these fractions as N tends to infinity. Note that (a) this limit exists not for all tilings; (b) if this limit exists, it does not change under shifts of the tiling. We restrict our attention to tilings in which this limit exists for each $P \subset \tau$. Similarly, for subsets $P,Q \subset \tau$ and a translation $t \in \mathbb{Z}^2$ we denote by $\nu(P \xrightarrow{t} Q)$ the limit fraction of points $x \in \mathbb{Z}^2$ such that $U(x) \in P$ and $U(x+t) \in Q$. We say that a tiling does not have long range order if probabilities of occurrences of tiles at distant sites are in some sence 'almost independent'. More precisely, we say that a τ-tiling U does not have long range order if for all $P,Q \subset \tau$

$$\lim_{|t|\to\infty} |\nu(P \xrightarrow{t} Q) - \nu(P)\cdot\nu(Q)| = 0 \qquad (*)$$

where $|t|$ is the usual Euclidean norm of vector t.

The following question was formulated by C. Radin in [11]: Does exist a tile set τ such that some τ-tilings exists and all τ-tilings do not have long range order? Obviously, such tilings must be aperiodic. However this requirement is much stronger than aperiodicity. In particular, the classic constructions of aperiodic tilings (e.g., [10, 1]) do not satisfy (*). In fact, (*) looks rather counterintuitive: we look for a local rule which guarantees that tiling must look globally "irregular" in some sense.

To the best of our knowledge, this question of Radin remains open. We believe that tilings without long range order can be obtained from self-similar tile set constructions based on the fixed-point argument. In this paper we explain the main ideas of this technique and prove a weaker version of the main conjecture:

Theorem 2 (main result)**.** *There exists a tile set τ that can be split into two disjoint parts $\tau = P \sqcup Q$ so that (a) τ-tilings exist, and (b) for each τ-tiling (*) is true.*

Since $\tau = P \sqcup Q$, we can correspond to each τ-tiling U a configuration in $U' \in \{0,1\}^{\mathbb{Z}^2}$: we let $U'(x) = 0$ if $U(x) \in P$ and $U'(x) = 1$ if $U(x) \in Q$. Theorem 2 claims that to each τ-tiling there corresponds a configuration U' satisfying (*). In other words, the local rules of τ admit tilings such that the natural projection $\pi : \tau^{\mathbb{Z}^2} \to \{0,1\}^{\mathbb{Z}^2}$ maps each τ-tiling to a configuration that does not have long range order.

In the rest of this paper we explain the idea of the fixed-point construction of tilings and give a sketch of the proof of Theorem 2.

2 Fixed point construction of a tiling

First we present a construction of an aperiodic tile set that is based on Kleene's fixed-point construction. This argument is similar to J. von Neumann self-reproducing automata [9]; very similar ideas were also used by P. Gács in the context of error-correcting computations [7]. This type of argument was used in [5] to construct tilings with some specific properties. Flexibility of this construction allows us to construct a tile set that is not only aperiodic but satisfies the claim of Theorem 2. We believe that the same technique can help answer the question of C. Radin in the most general form.

2.1 Macro-tiles

Let us fix some tile set τ and some integer $N > 1$. We call by a *macro-tile* any $N \times N$ square tiled by τ-tiles. Neighbor tiles in this square must match (adjacent sides must have equal colors); there is

no constraints for colors on the boarder line of this square. Thus, every side of a macro-tile carries a sequence of N colors. We call this sequence of colors *a macro-color*.

Let ρ be a set of τ-macro-tiles. We say that τ *simulates* ρ if (a) at least one τ-tilings exist, and (b) for each τ-tiling there exists a unique grid of vertical and horizontal lines that cuts this tiling into $N \times N$ macro-tiles from ρ.

For example, let a set ρ consist of exactly one macro-tile (that has the same macro-colors on all four sides) is simulated by some tile set τ. The tile set τ consists of N^2 tiles indexed by pairs (i, j) of integers modulo N. A tile from τ has colors on its sides as shown on Fig. 1. The macro-tile in ρ has colors $(0,0),\dots,(0,N-1)$ and $(0,0),\dots,(N-1,0)$ on its borders (Fig. 2).

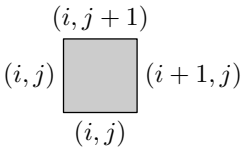


Figure 1: Basic tiles

If a tile set τ simulates some set ρ of τ-macro-tiles with zoom factor $N > 1$ and ρ is isomorphic to τ, then the set τ is called *self-similar*. We call by an *isomorphism* between τ and ρ some bijection that respects the relations "one tile can be placed on the right of another one" and "one tile can be placed on the top of another one". For the sake of brevity we say that a tile set τ simulates a tile set ρ when τ simulates some set of macro tiles $\tilde{\rho}$ isomorphic to ρ. So, a self-similar tile set simulates itself.

The idea of self-similarity was used (sometimes implicitly) in most constructions of aperiodic tile sets. Probably only the constructions in [8, 3] are the exceptions. The proof is based on a simple lemma:

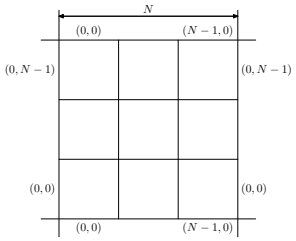


Figure 2: Macro-tile of size $N \times N$

Lemma 1 (folklore)**.** *A self-similar tile set τ has only aperiodic tilings.*

Thus, to prove that aperiodic tile set exits, it is enough to construct a self-similar tile set. We show how to construct such a tile set using the idea of Kleene's fixed-point theorem. To this end, we first explain how to simulate a given tile set by embedding computations (e.g., a Turing machine) in a tiling.

2.2 Simulating a tile set

We start with an informal discussion. Assume that we have a tile set ρ whose colors can be encoded as k-bit strings (i.e., $C \subset \mathbb{B}^k$) and the set of tiles $\rho \subset C^4$ is presented as a predicate $R(c_1, c_2, c_3, c_4)$. Assume that we have some Turing machine $\mathscr{R}$ that computes R. Now we show how to simulate ρ using some other tile set τ.

Our construction extends the example on Fig. 2. We keep the coordinate system modulo N embedded into tiles of τ. These coordinates guarantee that all τ-tilings can be uniquely cut into blocks of size $N \times N$ and every tile "knows" its position in the block. In addition to the coordinate system, now each tile in τ carries supplementary colors on its sides. On the border of a macro-tile only two supplementary colors (e.g., 0 and 1) are allowed. So the macro-color encodes a string of N bits, where N is the size of macro-tiles. We assume that $N \geq k$ and let k bits in the middle of macro-tile sides represent colors from C. All other bits on the sides are zeros (this is a restriction on the tile set: each tile knows its coordinates so it also knows whether non-zero supplementary colors are allowed).

Now we introduce additional restrictions on tiles in τ that guarantee that the macro-colors on sides of each macro-tile satisfy the relation R. To this end we ensure that bits from the macro-tile sides are transferred to the central part of the tile where the checking computation of $\mathscr{R}$ is simulated.

For this "information transmission" we fix which tiles in a macro-tile form "wires". This choice can be done in any reasonable way; we may even assume that wires do not cross each other. We require that tiles along each of these wires carries equal bits on two sides (so one bit value of is transmitted along a wire); again it is easy to organize since each tile knows its coordinates.

To check that the property R is true for a quadruple of macro-colors, we require that the central part of a macro-tile represents a time-space diagram of $\mathscr{R}$'s computation. Embedding of a space-time diagram in a tiling is a standard trick. We require that horizontal rows represent configurations of the tape of the Turing machine, time goes upwards. We require that computation terminates in an accepting state: if not, the tiling cannot be formed.

To make this construction work, the size of macro-tile should be large enough: we need enough space for k bits to propagate and enough time and space to perform accepting computation of $\mathscr{R}$.

In this construction the number of supplementary colors depends on the machine $\mathscr{R}$. The more states it has, the more colors are needed in the computation zone. To get rid of this dependency, we replace the ad hoc machine $\mathscr{R}$ by a fixed universal Turing machine $\mathscr{U}$ that runs a *program* simulating $\mathscr{R}$.

Talking about the universal Turing machine, we may require that the tape has an additional read-only layer. Each cell of this layer carries one bit that is not changed during the computation; these bits are used as a program for the universal machine (Fig. 3). So in the computation zone the columns carry unchanged bits, and the tile set restrictions guarantee that these bits form the program for $\mathscr{U}$, and the central zone represents the protocol of an accepting computation for that program. In this way we get a tile set τ that simulates ρ with zoom factor N using $O(N^2)$ tiles.

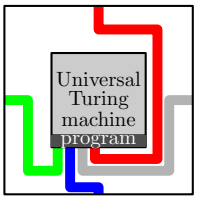


Figure 3: Scheme of a macro-tile

2.3 Simulating itself

We already know how to simulate a given tile set ρ (represented as a program for the universal Turing machine) by another tile set τ with a large enough zoom factor N. Now we want τ to be isomorphic to ρ itself. Here we use a construction that follows the fixed-point argument from Kleene's recursion theorem. Recall the construction from Section 2.2. Note that most rules of τ do not depend at all on the program for $\mathscr{R}$. They deal only with information transfer along the wires, the vertical propagation of unchanged program bits, and the space-time diagram for the universal Turing machine in the computation zone. We make these rules a part of ρ's definition, and get a program which checks that macro-tiles behave like τ-tiles in this respect.

The only remaining part of the rules for τ is the hardwired program. We must guarantee that macro-tiles carry the same program as τ-tiles do. Hence, our program for the universal Turing machine needs to access the bits of its own text. This self-referential action is quite legal. Indeed, the program is written on the tape, and the machine can read it (this argument is the key idea of the standard proof of Kleene's recursion theorem). The program checks that if a macro-tile belongs to the first line of the computation zone, this macro-tile carries the correct bit of the program.

Now we choose N. We need it to be large enough so the described above computations can fit in the computation zone. The used computations are rather simple. They all are polynomial in the input size, i.e., of size $O(\log N)$. So for large N it easily fits in $O(N)$ available space and time. This finishes the basic construction of a self-similar aperiodic tile set. This construction together with Lemma 1 is already enough to get Theorem 1. However to prove Theorem 2 we need to extend our self-referential technique.

3 Variable zoom factor

In our basic construction the macro-tiles of all levels are of the same size: each of them contained $N \times N$ macro-tiles of the previous level for some constant zoom factor N. It is not enough to achieve our main result, since we will need to host arbitrarily long computations in high-level macro-tiles. So we need an increasing sequence of zoom factors $N_0, N_1, N_2, \ldots$; macro-tiles of the first level are blocks of $N_0 \times N_0$ tiles; macro-tiles of the second level are blocks of $N_1 \times N_1$ macro-tiles of level 1 (and have size of

$N_0N_1 \times N_0N_1$ if measured in individual tiles). In general, macro-tiles of level k are made of $N_{k-1} \times N_{k-1}$ macro-tiles of level $k-1$ and have side $N_0N_1 \dots N_{k-1}$ measured in individual tiles.

However, all these macro-tiles (of all levels) still carry the same program in their computation zone. The difference between their behavior is caused by the input data on the tape: each macro-tiles "knows" its level since it is provided as a sequence of bits on its tape. Then this level k may be used to compute N_k which is then used as a modulus for coordinates in the father macro-tile.

We must ensure that this information is correct. We need two properties: (a) all macro-tiles of the same level have the same idea about their level, and (b) these ideas are consistent between levels (each father is one level higher than its sons). To achieve these properties we involve additional features in our macro-tiles. Since each macro-tile knows its position inside the father, so it knows whether the father should keep some bits of his level information in that macro-tile. If yes, the macro-tile checks that this information is correct. Each macro-tile checks only one bit of the level information, but with brothers' help they check all the bits in their father's memory.

Also we should check that the level information fits into the tiles, and the computation required to compute N_k from k also fits into level k tile. This means that $\log k$, $\log N_k$ and the time needed to compute N_k from k should be much less than N_{k-1} (since the computation zone is some fraction of N_{k-1}). So N_k should not grow very slow (e.g., $N_k = \log k$ is too slow), should not grow very fast (e.g., $N_k = 2^{N_{k-1}}$ is too fast) and should not be too hard to compute. All these restriction still leave us a lot of freedom. E.g., N_k can be proportional to $\sqrt{k}$, to k, to 2^k, etc. In the next section we assume that $N_k = 2^{C^k}$ from some large enough constant C. Now computational zones of macro-tiles increase on each next level, so we can embed there computations of increasing sizes into the macro-tiles.

4 Sketch of the proof of Theorem 2

In our basic construction every tile knows its coordinates in the macro-tile and some additional information needed to arrange 'wires' and simulate calculations of the universal Turing machine. Now in addition to this basic structure each tile keeps one *auxiliary bit* of information. Formally it means that the set of all tiles is split into two parts: $\tau = \tau_0 \sqcup \tau_1$. These bits (assigned to all tiles of a tiling) will make a configuration in $\{0,1\}^{\mathbb{Z}^2}$ that satisifies (*). Similarly, we assign an *auxiliarely bit* to each macro-tile (i.e., on each level we will have 0-type or 1-type macro-tiles).

On each level k, we allow exactly two different valid distributions of auxiliary bits in a k-level macro-tiles; these two distributions will be opposite to each other (one is the "reversed image" of the other one). For each k-level macro-tile the choice between these two possible distributions of subordinate "auxiliary bits" is determined by the "auxiliary bit" assigned to this macro-tile itself. The two valid distributions of auxiliary bits inside a k-level macro-tile are defined inductively: first we chose two valid distributions of 0-tiles and 1-tiles in a 1-level macro-tile of size $N_0 \times N_0$ (there will be exactly 50% of 0-tiles and 50% of 1-tiles in a macro-tile of level 1); then we define two valid distributions of 1-level macro-tile in a 2-level macro-tile (which is an $N_1 \times N_1$-matrix of 1-level macro-tiles), etc.

The distribution of auxiliary bits inside of a k-level macro-tile should be chosen in such a way that for every shift t such that $N_0N_1 \dots N_{k-1} \le |t| < N_0N_1 \dots N_k$ the difference $|\nu(P \xrightarrow{t} Q) - \nu(P) \cdot \nu(Q)|$ is bounded by $1/k$ (so, it tend to 0 as k tends to infinity). The fact that such configurations of auxiliary bits exist, follows from a simple probabilistic argument. In what follows, we explain this argument in some detail.

Assume we already fixed valid distributions of 0-tiles and 1-tiles in all macro-tiles of level up to k. Now we decide how to distribute k-level macro-tiles of type 0 and 1 inside a matrix of size $N_k \times N_k$, which makes a $(k+1)$-level macro-tile. Let us draw independent random bits for each row and each column of a $(k+1)$-level macro-tile (i.e., we need $2N_k$ random bits), and assign to each macro-tile of

level k inside the $N_k \times N_k$-matrix XOR of the two random bits corresponding to its row and column.

Now consider any $1 \times N_k$ row of k-level macro-tiles in the constructed macro-tile of level $k+1$ (this "row" is a rectangle of size $(N_0 \cdot \ldots \cdot N_{k-1}) \times (N_0 \cdot \ldots \cdot N_k)$ paved by τ-tiles), and consider the shift of this row by vector t. It is not hard to check (with standard Chernoff bounds) that with very high probability this shift "changes" assigned bits for approximately 50% of all tiles in the block. The typical deviation of the fraction of "changed" bits from 50% is very small and can be bounded by $1/k$. Technically the deviation of this frequency is greater than $1/k$ only with probability at most $e^{-\Omega(N_k/\mathrm{poly}(k))} \cdot \mathrm{poly}(N_k)$. (Remark: this bound requires the assumption $N_{k-1} \ll N_k$. For the lack of space we skip the calculations.) Probability of large deviations remains very small even when we sum up it for all rows in a $(k+1)$-level macro-tile (i.e., multiply it by N_k).

Thus, we proved probabilistically that the required distribution of k-level macro-tiles inside a $(k+1)$-level macro-tile exists. How to enforce such a coloring by local rules? First of all, we can guarantee that this distributions are the same (or exactly opposite) in every two neighboring $(k+1)$-level macro-tiles (in our construction the choice of the distribution of 0- and 1-tiles in a $(k+1)$-level macro-tile is determined by only $O(N_k)$ bit parameters; so this information can be passed through the borderline of neighboring macro-tiles of size $N_k \times N_k$). It remains to guarantee that the valid distribution of k-level macro-tiles indeed implies that shifts t such that $N_0 N_1 \ldots N_{k-1} \le |t| < N_0 N_1 \ldots N_k$ changes about 50% of "auxiliary bits" inside every $(k+1)$-level macro-tile. To this end, the computational zone of each $(k+2)$-level macro-tile checks the required property is true. We have enough room for this computation since $N_{k+1} \gg N_k$. To run this computation we need to deliver to the computational zone of every $(k+2)$-level macro-tile the auxiliary bits assigned to its subordinate macro-tiles of lower levels. This can be done by the bit propagation technique from [6]. Since the complete argument is rather technical, we have to skip detail for the lack of space.

5 Further research

We see two main avenues for the subsequent work. First, we believe that our technique is suitable to answer the question of C. Radin in the most general setting. Another intriguing question is how to implement the fixed-point construction of a tile set with a smaller size of a tile set. In the present argument the number of different tiles in a tile set is about several millions. It would be interesting to obtain a similar construction with hundreds or even dozens of tiles. This probably requires involving other programming model, supporting more succinct programming style than Turing machines.

References

[1] C. Allauzen, B. Durand. Appendix A: Tiling Problems. In: E. Börger, E. Grädel, Y. Gurevich, *The Classical Decision Problems*, Springer-Verlag, 1996.

[2] R. Berger, The Undecidability of the Domino Problem. *Mem. Amer. Math. Soc.*, **66**, 1966.

[3] K. Culik, An Aperiodic Set of 13 Wang Tiles, *Discrete Math.*, **160**, 245–251, 1996.

[4] B. Durand, L. Levin, A. Shen, Complex Tilings. *J. Symbolic Logic*, **73** (2), 593–613, 2008 (See also Proc. *33rd Ann. ACM Symp. Theory Computing*, 732–739, 2001.

[5] B. Durand, A. Romashchenko, A. Shen, Fixed-point tile sets and their applications. arXiv:0910.2415.

[6] B. Durand, A. Romashchenko, A. Shen, Effective closed subshifts in 1D can be implemented in 2D. arXiv:1003.3103

[7] P. Gács, Reliable Cellular Automata with Self-Organization, *J. Stat. Phys.*, **103** (1/2), 45–267, 2001.

[8] J. Kari, A Small Aperiodic Set of Wang tiles, *Discrete Math.*, **160**, 259–264, 1996.

[9] J. von Neumann, *Theory of Self-reproducing Automata*, Edited by A. Burks, Univ. of Illinois Press, 1966.

[10] R. Robinson, Undecidability and Nonperiodicity for Tilings of the Plane. *Inventiones Mathematicae*, **12**, 177–209, 1971.

[11] C. Radin, *Miles of tiles*. Student Mathematical Library. AMS, 1994.

A note on strong dinaturality, initial algebras and uniform parameterized fixpoint operators

Tarmo Uustalu
Institute of Cybernetics at Tallinn University of Technology,
Akadeemia tee 21, EE-12618 Tallinn, Estonia
tarmo@cs.ioc.ee

Abstract

I show a basic Yoneda-like lemma relating strongly dinatural transformations and initial algebras. Further, I apply it to reprove known results about unique existence of uniform parameterized fixpoint operators.

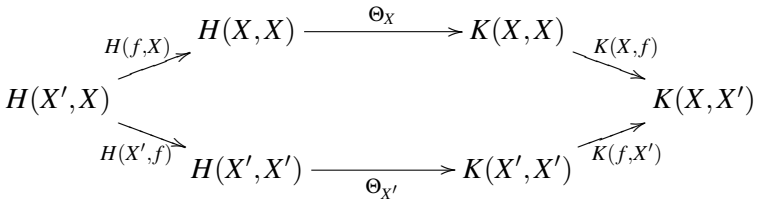
1 Introduction

I present a Yoneda-like lemma relating strongly dinatural (a.k.a. Barr dinatural) transformations and initial functor-algebras. It is very basic, but I do not know whether it has appeared in the literature. I have found it quite useful: it can be used, for example, to prove the validity of some Mendler-style structured recursion schemes for initial algebras or recursive coalgebras [12, 14] and to prove properties of Church representations of inductive types [9, 5]. Here, I use it to reprove some known results [7, 10] about existence and unique existence of uniform parameterized fixpoint operators, exploiting that uniformity is a strong dinaturality condition. I would not dare to claim that the proofs become simpler, but they obtain a structure that nicely localizes the invocations of the various initial and bifree algebra existence assumptions made.

2 Strong dinaturality and a Yoneda lemma for initial algebras

Dinatural transformations [2] and strongly dinatural (a.k.a. Barr dinatural) transformations [7, 8] are two generalizations of natural transformations from (covariant) functors to mixed-variant functors that have components only defined for the diagonal of the domain. We recall the definitions, starting with dinaturality.

Definition 1 (Dinaturality)**.** *A* dinatural transformation *between* $H,K \in \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{E}$ *is given by, for any* $X \in |\mathbb{C}|$*, a map* $\Theta_X \in \mathbb{E}(H(X,X),K(X,X))$ *such that, for any* $f \in \mathbb{C}(X,X')$*, the following hexagon commutes in* $\mathbb{E}$*:*

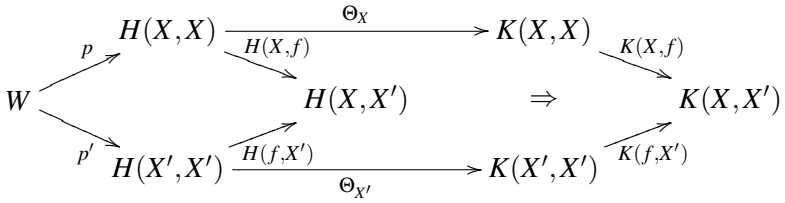


Dinatural transformations are used, for example, in the definitions of coend and end. A coend is an initial cowedge, where a cowedge is given by an object and an accompanying dinatural transformation (just as a colimit is defined as an initial cocone, a cocone being an object with a natural transformation).

Dinatural transformations do not generally compose and so do not give a category. Strongly dinatural transformations do not suffer from this shortcoming.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 77-82

Definition 2 (Strong dinaturality)**.** *A* strongly dinatural transformation *between* $H, K \in \mathbb{C}^{\mathrm{op}} \times \mathbb{C} \to \mathbb{E}$ *is given by, for any* $X \in |\mathbb{C}|$*, a map* $\Theta_X \in \mathbb{E}(H(X,X), K(X,X))$ *such that, for any map* $f \in \mathbb{C}(X,X')$ *and any span* (W,p,p') *on* (X,X')*, if the square in the following diagram commutes in* $\mathbb{E}$*, then so does the hexagon:*



If $\mathbb{E}$ is a category with pullbacks such as, e.g., **Set**, one can equivalently require that, for every map $f \in \mathbb{C}(X,X')$, the outer hexagon of the above diagram commutes for (W,p,p') the chosen pullback of the cospan $(H(X,X'), H(X,f), H(f,X'))$.

We write $[\mathbb{C},\mathbb{E}]_{\mathrm{sd}}$ for the category of mixed-variant functors from $\mathbb{C}$ to $\mathbb{E}$ and strongly dinatural transformations.

Any strongly dinatural transformation is also dinatural, but the converse does not hold in general.

This note is centered around the following observation, which I have not noticed published (I have mentioned it in an unpublished talk abstract [11] ten years ago, and also in a paper on the recursion scheme from the cofree recursive comonad [14]). Please be so kind and tell me, if you know of a reference where it might appear. It is a kind of a Yoneda lemma for strongly dinatural transformations and initial algebras.

Proposition 1 (Yoneda lemma for initial algebras)**.** *Let* $\mathbb{C}$ *be a locally small category,* $F \in \mathbb{C} \to \mathbb{C}$ *a functor with an initial algebra (which we denote* $(\mu F, \mathsf{in}_F)$*) and* $K \in \mathbb{C} \to \mathbf{Set}$ *a functor (whose padding into a mixed-variant functor we denote also by* K*). Then*

$$[\mathbb{C},\mathbf{Set}]_{\mathrm{sd}}(\mathbb{C}(F-,-),K) \cong K(\mu F)$$

(so $[\mathbb{C},\mathbf{Set}]_{\mathrm{sd}}(\mathbb{C}(F-,-),K)$ *is, in fact, a set too). This isomorphism is natural in* F *(to the extent that initial algebras exist in* $\mathbb{C}$*).*

A strongly dinatural transformation between $\mathbb{C}(F-,-)$ and K is given by, for any X, a map $\Theta_X \in \mathbb{C}(FX,X) \to KX$, such that, for any X, X', $\phi \in \mathbb{C}(FX,X)$, $\phi' \in \mathbb{C}(FX',X')$, $f \in \mathbb{C}(X,X')$, $f \circ \phi = \phi' \circ Ff$ (i.e., f being an F-algebra map from between (X,ϕ) and (X',ϕ')) implies $\Theta_{X'} = Kf\,\Theta_X \in KX'$.

We denote the natural isomorphism by i_F. It is defined as follows: for $\Theta \in [\mathbb{C},\mathbf{Set}]_{\mathrm{sd}}(\mathbb{C}(F-,-),K)$, $\mathsf{i}_F\,\Theta =_{\mathrm{df}} \Theta_{\mu F}\mathsf{in}_F \in K(\mu F)$; and, for $x \in K(\mu F)$, $X \in |\mathbb{C}|$, $k \in \mathbb{C}(FX,X)$, $(\mathsf{i}_F^{-1}\,x)_X\,k =_{\mathrm{df}} K(\mathsf{fold}_{F,X}\,k)\,x \in KX$, where $\mathsf{fold}_{F,X}$ denotes the unique algebra map from $(\mu F,\mathsf{in}_F)$ to (X,k).

An important special case is when $KX =_{\mathrm{df}} \mathbb{C}(1,X)$. We get that

$$[\mathbb{C},\mathbf{Set}]_{\mathrm{sd}}(\mathbb{C}(F-,-),\mathbb{C}(1,-)) \cong \mathbb{C}(1,\mu F)$$

This is closely related to Church representations of inductive types. Remember that, in System F, we represent μF by $\forall X.\,(F\,X \Rightarrow X) \Rightarrow X)$.

Needless to say, for final coalgebras, a dual proposition is true; I refrain from spelling it out here.

3 Uniform parameterized fixpoint operators

We now turn to parameterized fixpoint-like operators and uniformity. The dependencies between different axiomatiozations and sufficient conditions for existence have been studied by Bloom and Ésik [1],

Freyd [3, 4], Mulry [7], Simpson and Plotkin [10] etc. Hyland and Hasegawa [6] showed that (uniform) Conway operators are equivalent to (uniform) traces (definable in general monoidal categories, not just categories with finite products).

We will closely follow the account of Simpson and Plotkin [10]. First we recall the definitions of parameterized fixpoint operators, parameterized Conway operators and uniformity.

We assume given a category $\mathbb{D}$ with finite products.

Definition 3 (Parameterized fixpoint-like operator)**.** *A* parameterized fixpoint-like operator *on* $\mathbb{D}$ *is given by, for any* $X, Y \in |\mathbb{D}|$*, a function* $\mathsf{fix}_{X,Y} \in \mathbb{D}(X \times Y, Y) \to \mathbb{D}(X, Y)$.

Definition 4 (Parameterized fixpoint operator)**.** *A* parameterized fixpoint operator *on* $\mathbb{D}$ *is a parameterized fixpoint-like operator* fix *on* $\mathbb{D}$ *such that*

- *for any* $f \in \mathbb{D}(X, X')$ *and* $k' \in \mathbb{D}(X' \times Y, Y)$, $\mathsf{fix}(k' \circ (f \times \mathsf{id}_Y)) = \mathsf{fix}\, k' \circ f$ (naturality)*;*
- *for any* $k \in \mathbb{D}(X \times Y, Y)$, $\mathsf{fix}\, k = k \circ \langle \mathsf{id}_X, \mathsf{fix}\, k \rangle$ (parameterized fixpoint property).

Definition 5 (Conway operator)**.** *A* Conway *operator on* $\mathbb{D}$ *is a parameterized fixpoint operator* fix *on* $\mathbb{D}$ *with the further properties that*

- *for any* $f \in \mathbb{D}(X \times Y, Y')$ *and* $h \in \mathbb{D}(X \times Y', Y)$, $f \circ \langle \mathsf{id}_X, \mathsf{fix}(h \circ \langle \mathsf{fst}, f \rangle) \rangle = \mathsf{fix}(f \circ \langle \mathsf{fst}, h \rangle)$ (parameterized dinaturality)*;*
- *for any* $k \in \mathbb{D}((X \times Y) \times Y, Y)$, $\mathsf{fix}(k \circ \langle \mathsf{id}_{X \times Y}, \mathsf{snd}_{X,Y} \rangle) = \mathsf{fix}(\mathsf{fix}\, k)$ (diagonal property).

Parameterized dinaturality implies the parameterized fixpoint property, so for Conway operators the latter condition is redundant.

For our final definition, we assume we also have a category $\mathbb{C}$ with finite products and the same objects as $\mathbb{D}$ together with an identity-on-objects functor $J \in \mathbb{C} \to \mathbb{D}$ preserving the finite products of $\mathbb{C}$ strictly. We call the maps of $\mathbb{D}$ in the image of J *pure*.

Definition 6 (Uniformity)**.** *A parameterized fixpoint-like operator* fix *on* $\mathbb{D}$ *is said to be* uniform *wrt.* J*, if*

- *for any* $f \in \mathbb{C}(Y, Y')$, $k \in \mathbb{D}(X \times Y, Y)$ *and* $k' \in \mathbb{D}(X \times Y', Y')$, $J f \circ k = k' \circ (\mathsf{id}_X \times J f)$ *implies* $J f \circ \mathsf{fix}\, k = \mathsf{fix}\, k'$.

(In the terminology of iteration theories [1], naturality is parameter identity, parameterized fixpoint property is fixpoint identity, parameterized dinaturality is composition identity and diagonal property is double dagger identity. Finally, uniformity corresponds to the functoriality condition.)

We now focus on the special case of $\mathbb{D}$ arising as the coKleisli category of a comonad $(D, \varepsilon, (-)^\dagger)$[1] on $\mathbb{C}$ with J the right adjoint in its coKleisli splitting. The prototypical well-behaved situation here has $\mathbb{C} =_{\mathrm{df}} \mathbf{Cppo}_\perp$, $D =_{\mathrm{df}} (-)_\perp$ and $\mathbb{D} \cong \mathbf{Cppo}$, where **Cppo** stands for the category of ω-complete pointed partial orders and ω-continuous functions, $\mathbf{Cppo}_\perp$ is as **Cppo** but has as maps only the strict (bottom-preserving) maps of **Cppo** and $(-)_\perp$ is the lifting endofunctor. More generally, the lifting comonad can be replaced with any comonad on $\mathbf{Cppo}_\perp$ that has its underlying functor **Cppo**-enriched.

In terms of the "base" category $\mathbb{C}$, a parameterized fixpoint-like operator is now, for any $X, Y \in |\mathbb{C}|$, a function $\mathsf{fix}_{X,Y} \in \mathbb{C}(D(X \times Y), Y) \to \mathbb{C}(DX, Y)$. The various optional additional conditions specialize into the following:

[1] We write $(-)^\dagger$ for the coKleisli extension operation of a comonad.

- for any $f \in \mathbb{C}(DX, X')$ and $k' \in \mathbb{C}(D(X' \times Y), Y)$, $\mathsf{fix}\,(k' \circ \langle f \circ D\,\mathsf{fst}, \varepsilon_Y \circ D\,\mathsf{snd}\rangle^\dagger) = \mathsf{fix}\,k' \circ f^\dagger$ (naturality);
- for any $k \in \mathbb{C}(D\,(X \times Y), Y)$, $\mathsf{fix}\,k = k \circ \langle \varepsilon_X, \mathsf{fix}\,k\rangle^\dagger$ (parameterized fixpoint property);
- for any $f \in \mathbb{C}(D\,(X \times Y), Y')$ and $h \in \mathbb{C}(D\,(X \times Y'), Y)$, $f \circ \langle \varepsilon_X, \mathsf{fix}\,(h \circ \langle \varepsilon_X \circ D\,\mathsf{fst}, f\rangle^\dagger)\rangle^\dagger = \mathsf{fix}\,(f \circ \langle \varepsilon_X \circ D\,\mathsf{fst}, h\rangle^\dagger)$ (parameterized dinaturality);
- for any $k \in \mathbb{C}(D\,((X \times Y) \times Y), Y)$, $\mathsf{fix}\,(k \circ D\,\langle \mathsf{id}_{X\times Y}, \mathsf{snd}_{X,Y}\rangle) = \mathsf{fix}\,(\mathsf{fix}\,k)$ (diagonal property);
- for any $f \in \mathbb{C}(Y, Y')$, $k \in \mathbb{C}(D\,(X \times Y), Y)$ and $k' \in \mathbb{C}(D\,(X \times Y'), Y')$, $f \circ k = k' \circ D\,(\mathsf{id}_X \times f)$ implies $f \circ \mathsf{fix}\,k = \mathsf{fix}\,k'$ (uniformity).

(Notice that here, id and $\circ$ refer to identity and composition in $\mathbb{C}$ rather than in $\mathbb{D}$, differently from what they meant above.)

Crucially for us, the uniformity condition asserts nothing else than strong dinaturality of $\mathsf{fix}_{X,Y}$ in Y, i.e., that $\mathsf{fix}_{X,-} \in [\mathbb{C}, \mathbf{Set}]_{\mathrm{sd}}(\mathbb{C}(D\,(X \times -), -), \mathbb{C}(DX, -))$—an observation first made by Mulry [7].

From Proposition 1, we immediately get:

Corollary 1. *If every functor $D(X \times -) \in \mathbb{C} \to \mathbb{C}$ has an initial algebra, then a uniform wrt. J parameterized fixpoint-like operator* fix *on $\mathbb{D}$ is the same as, for any $X \in |\mathbb{C}|$, a map $\underline{\mathsf{fix}}_X \in \mathbb{C}(DX, \mu\,(D\,(X \times -)))$.*

The bijection is given by $\underline{\mathsf{fix}}_X =_{\mathrm{df}} \mathsf{fix}_{X,\mu(D(X\times -))}\,\mathsf{in}_{D(X\times -)}$ and, for $k \in \mathbb{C}(D(X \times Y), Y)$, $\mathsf{fix}_{X,Y}\,k =_{\mathrm{df}} \mathsf{fold}_{D(X\times -),Y}\,k \circ \underline{\mathsf{fix}}_X$.

It is not difficult to strengthen this corollary to the following characterization of uniform parameterized fixpoint operators (one has to verify that the conditions are pairwise equivalent):

Proposition 2. *If every functor $D\,(X \times -) \in \mathbb{C} \to \mathbb{C}$ has an initial algebra, then a uniform wrt. J parameterized fixpoint operator* fix *on $\mathbb{D}$ is the same as, for any $X \in |\mathbb{C}|$, a map $\underline{\mathsf{fix}}_X \in \mathbb{C}(DX, \mu\,(D\,(X \times -)))$ such that*

- *for any $f \in \mathbb{C}(DX, X')$, $\mu\,(\langle f \circ D\,\mathsf{fst}, \varepsilon_- \circ D\,\mathsf{snd}\rangle^\dagger) \circ \underline{\mathsf{fix}}_X = \underline{\mathsf{fix}}_{X'} \circ f^\dagger$ ("naturality");*
- *for any $X \in |\mathbb{C}|$, $\underline{\mathsf{fix}}_X = \mathsf{in}_{D(X\times -)} \circ \langle \varepsilon_X, \underline{\mathsf{fix}}_X\rangle^\dagger$ ("parameterized fixpoint property").*

Recall that a bifree algebra is an initial algebra that is at the same time also a final coalgebra. The following is nearly immediate from the proposition we just stated.

Proposition 3 ([10, Proposition 6.5])**.** *If every functor $D(X \times -) \in \mathbb{C} \to \mathbb{C}$ has a bifree algebra, then $\mathbb{D}$ has a unique uniform wrt. J parameterized fixpoint operator.*

Proof. Just observe that the parameterized fixpoint property can be rewritten as $\mathsf{in}^{-1}_{D(X\times -)} \circ \underline{\mathsf{fix}}_X = D\,(X \times \underline{\mathsf{fix}}_X) \circ \langle \varepsilon_X, \mathsf{id}_{DX}\rangle^\dagger$, which stipulates that $\underline{\mathsf{fix}}_X$ (if existing) must be a coalgebra map between $(DX, \langle \varepsilon_X, \mathsf{id}_{DX}\rangle^\dagger)$ and $(\mu\,(D\,(X \times -)), \mathsf{in}^{-1}_{D(X\times -)})$. As the latter is a final coalgebra, there is exactly one such map. This map turns out to also satisfy the required naturality condition. □

Uniform Conway operators can be analyzed similarly. Here we need the existence of further initial algebras to replace conditions on fix with conditions on $\underline{\mathsf{fix}}$. When these initial algebras are also final coalgebras, we have a unique uniform Conway operator.

Proposition 4. *If all functors $D\,(X \times -), D\,(X \times D\,(X \times -)), D\,((X \times -) \times -) \in \mathbb{C} \to \mathbb{C}$ have initial algebras, then a uniform wrt. J Conway operator on $\mathbb{D}$ is the same as, for any $X \in |\mathbb{C}|$, a map $\underline{\mathsf{fix}}_X \in \mathbb{C}(DX, \mu\,(D\,(X \times -)))$ satisfying the conditions of Proposition 2, but also the following conditions:*

- *for any* $X \in |\mathbb{C}|$, $\mathsf{in} \circ \langle \varepsilon_X, \mathsf{fold}\,(\langle \varepsilon_X \circ D\,\mathsf{fst}, \mathsf{in}\rangle^\dagger) \circ \underline{\mathsf{fix}}\rangle^\dagger = \mathsf{fold}\,(\mathsf{in} \circ \langle \varepsilon_X \circ D\,\mathsf{fst}, \mathsf{id}\rangle^\dagger) \circ \underline{\mathsf{fix}} \in \mathbb{C}(DX, \mu(D(X \times D(X \times -))))$ *("parameterized dinaturality");*

- *for any* $X \in |\mathbb{C}|$, $\mathsf{fold}\,(\mathsf{in} \circ D\,\langle \mathsf{id}, \mathsf{snd}\rangle) \circ \underline{\mathsf{fix}} = \mathsf{fold}\,(\mathsf{fold}\,\mathsf{in} \circ \underline{\mathsf{fix}}) \circ \underline{\mathsf{fix}} \in \mathbb{C}(DX, \mu(D((X \times -) \times -)))$ *"diagonal property").*

Proposition 5 ([10, Theorem 3])**.** *If all functors* $D(X \times -), D(X \times D(X \times -)), D((X \times -) \times -) \in \mathbb{C} \to \mathbb{C}$ *have bifree algebras, then* $\mathbb{D}$ *has a unique uniform wrt. J Conway operator.*

4 Uniform guarded recursion operators

A similar treatment is possible for guarded recursion operators (we have previously considered some aspects for the dual situation of guarded iteration [13]). Here, the prototypical example is given by cofree recursive comonads on endofunctors on **Set**, such as the nonempty list comonad defined by $DX =_{\mathrm{df}} \mu(X \times (1 + (-)))$.

An ideal comonad on a category $\mathbb{C}$ with finite products is a comonad given by $DX =_{\mathrm{df}} X \times D_0 X$, $\varepsilon_X =_{\mathrm{df}} \mathsf{fst} \in \mathbb{C}(DX, X)$, for any $k \in \mathbb{C}(DX, Y)$, $k^\dagger =_{\mathrm{df}} \langle k, k^\ddagger \circ \mathsf{snd}\rangle \in \mathbb{C}(DX, DY)$ where D_0 is an endofunctor on $\mathbb{C}$ and, for any $X, Y \in |\mathbb{C}|$, $(-)^\ddagger_{X,Y} \in \mathbb{C}(DX, Y) \to \mathbb{C}(D_0 X, D_0 Y)$.

A guarded recursion operator for an ideal comonad is, for any $X, Y \in |\mathbb{C}|$, a unique function $\mathsf{rec}_{X,Y} \in \mathbb{C}(X \times D_0(X \times Y), Y) \to \mathbb{C}(DX, Y)$ satisfying the guarded recursion equation $\mathsf{rec}\,k = k \circ (\mathsf{fst} \times \mathsf{id}) \circ \langle \varepsilon, \mathsf{rec}\,k\rangle^\dagger$ and possibly further properties.

As soon as all functors $X \times D_0(X \times -)$ have initial algebras, having a uniform guarded recursion operator rec becomes equivalent to having, for any $X \in |\mathbb{C}|$, a map $\underline{\mathsf{rec}}_X \in \mathbb{C}(DX, \mu(X \times D_0(X \times -))$ such that $\underline{\mathsf{rec}} = \mathsf{in} \circ (\mathsf{fst} \times \mathsf{id}) \circ \langle \varepsilon, \underline{\mathsf{rec}}\rangle^\dagger$.

A uniform guarded recursion operator exists uniquely, e.g., whenever D is the cofree recursive comonad on an endofunctor H on $\mathbb{C}$, in which case $DX \cong \mu(X \times H(-)) \cong X \times \mu(H(X \times -))$.

5 Conclusion

I find it intruiging that the use of the Yoneda-like lemma stages the invocations of the initial algebra resp. bifree algebra existence assumptions: the initial algebra existence assumptions ensure the possibility of reducing the existence of a parameterized fixpoint operator to the existence of a family of maps to initial algebras; the bifree algebra existence assumptions ensure that such a family of maps exists uniquely.

I would like to learn more about the relationship of strong dinaturality and models of parametric polymorphism.

Acknowledgment I thank my anonymous referees for useful remarks. This research was supported by the Estonian Science Foundation under grant no. 6940.

References

[1] S. Bloom, Z. Ésik. *Iteration theories. EATCS Monographs on Theor. Comput. Sci.*. Springer, 1993.

[2] E. Dubuc, R. Street. Dinatural transformations. In S. Mac Lane, ed., *Reports of the Midwest Category Seminar IV*, v. 137 of Lect. Notes in Math., pp. 126–137. Springer, 1970.

[3] P. Freyd. Algebraically complete categories. In A. Carbone, M. C. Pedicchio, G. Rosolini, eds., *Proc. of 1990 Int. Conf. on Category Theory (Como, July 1990)*, v. 1488 of *Lect. Notes in Math.*, pp. 95–104. Springer, 1991.

[4] P. Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P. T. Johnstone, A. M. Pitts, eds., *Applications of Categories in Computer Science*, v. 177 of *LMS Lect. Note Series*, pp. 95–106. Cambridge Univ. Press, 1992.

[5] N. Ghani, T. Uustalu, V. Vene. Build, augment and destroy, universally. In W.-N. Chin, ed., *Proc. of 2nd Asian Symp. on Programming Languages and Systems, APLAS 2004 (Taipei, Nov. 2004)*, v. 3302 of *Lect. Notes in Comput. Sci.*, pp. 327–347. Springer, 2004.

[6] M. Hasegawa. The uniformity principle on traced monoidal categories. *Publ. of RIMS, Kyoto Univ.*, 40(3):991–10143, 2004.

[7] P. S. Mulry. Strong monads, algebras and fixed points. In M. P. Fourman, P. T. Johnstone, A. M. Pitts, eds., *Applications of Categories in Computer Science*, v. 177 of *LMS Lect. Note Series*, pp. 202–216. Cambridge Univ. Press, 1992.

[8] R. Paré, L. Roman. Dinatural numbers. *J. of Pure and Appl. Algebra*, 128(1):33–92, 1998.

[9] D. Pavlovic. Logic of build fusion. Tech. rep. KES.U.00.9, Kestrel Institute, 2000.

[10] A. Simpson, G. Plotkin. Complete axioms for categorical fixed-point operators. In *Proc. of 15th Ann. IEEE Symp. on Logic in Computer Science, LICS 2000 (Santa Barbara, CA, June 2000)*, pp. 30–41. IEEE CS Press, 2000.

[11] T. Uustalu. Strong dinaturality and initial algebras (abstract). In *Abstracts of 12th Nordic Wksh. on Programming Theory, NWPT 2000 (Bergen, Oct. 2000)*, 2 pp. Univ. i Bergen, 2000. http://www.ii.uib.no/~nwpt00/Proceedings/Uustalu-abs.ps

[12] T. Uustalu, V. Vene. Coding recursion à la Mendler (extended abstract). In J. Jeuring, ed., *Proc. of 2nd Wksh. on Generic Programming, WGP 2000 (Ponte de Lima, July 2000)*, Tech. rep. UU-CS-2000-19, pp. 69–85. Dept. of Comput. Sci., Utrecht Univ., 2000.

[13] T. Uustalu, V. Vene. An alternative characterization of complete iterativeness (extended abstract). In Z. Ésik, I. Wałukiewicz, eds., *Proc. of 5th Int. Wksh. on Fixed Points in Computer Science, FICS 2003 (Warsaw, Apr. 2003)*, pp. 81–83. Warsaw Univ., 2003.

[14] T. Uustalu, V. Vene. The recursion scheme from the cofree recursive comonad. In V. Capretta, C. McBride, eds., *Proc. of 2nd Wksh. on Mathematically Structured Functional Programming, MSFP 2008 (Reykjavik, July 2008)*, *Electron. Notes in Theor. Comput. Sci.*, Elsevier, to appear.

Common patterns for metric and ordered fixed point theorems

Paweł Waszkiewicz*
Jagiellonian University, Kraków, Poland
pqw@tcs.uj.edu.pl

Abstract

We show that: (a) Banach's and Knaster-Tarski's fixed point theorems are instances of a single theorem with a constructive proof; (b) Bourbaki-Witt's and Caristi's theorems are instances of a single theorem that cannot have a constructive proof.

1 Introduction

Banach's and Knaster-Tarski's fixed point theorems have constructive proofs. In the first part of this paper we show that more is true: both statements are *instances* of a single theorem. As we shall see, in both cases the fixed point of a given map $f\colon X \to X$ is found by examining its generalized direct image f^* on a suitably constructed directed-complete poset. Since f^* is always order-preserving, regardless of the properties of f, we can apply the well-known theorem that an order-preserving map on a pointed dcpo has the least fixed point; remarkably, as shown by D. Pataraia [12], this statement has a fully constructive proof, formalizable in higher-order intuitionistic logic (in 2003 it found an entry to the compendium on continuous lattices and domains [7], where it is presented as a set of exercises). Hence we are going to use Pataraia's construction to simultaneously prove Banach's and Knaster-Tarski's theorems.

In the second part of this paper we show that Caristi's [3] theorem is precisely the metric analogue of the Bourbaki-Witt theorem. Again, both results are instances of a single statement, but this time we do not use any external help in the proof — it is the Bourbaki-Witt construction that proves the unifying theorem. Now, it has recently been discovered by A. Bauer [1] that the Bourbaki-Witt theorem can never be demonstrated by a fully constructive argument. In what follows we are going to use Bauer's insight to show that unless a proof of Caristi's theorem substantially rely on the symmetry axiom of the metric, then it cannot be accepted intuitionistically.

1.1 A construction of fixed points for order-preserving maps on dcpos

Since we will refer to details of Pataraia's proof, we shall start with a concise description of his construction.

Recall that a map $f\colon P \to P$ on a poset P is *expanding* if $x \leqslant fx$, for all x. We say that an order-preserving expanding map is *inflationary*. A *dcpo* is a poset P where each directed set has a supremum. A poset is *pointed*, if it has the least element, usually denoted as $\bot$. Now, suppose f is order-preserving on a pointed dcpo P. Following Pataraia's line of thought, we look for subsets of P that (a) contain $\bot$, (b) are closed under f, and (c) are directed-complete. Clearly $Y := \{x \in P \mid x \leqslant fx\}$ is one of them. Let C be the intersection of all sets with (a)-(c). Therefore, $f\colon C \to C$ is inflationary. Now, the set $E(C)$ of all inflationary maps on C, ordered pointwise, is directed-complete, and — and this is the crux of the construction — it is itself a directed set. The reason is that for $g,h \in E(C)$, we have $g,h \leqslant g \circ h$. Hence $E(C)$ has a top element $m\colon C \to C$. Consequently, $f \circ m = m$, and thus $f(m(\bot)) = m(\bot)$, i.e. $m(\bot) \in C$ is a fixed point of f. If $x \in P$ is some other fixed point of f, then $\downarrow x := \{y \in P \mid y \leqslant x\}$ satisfies (a)-(c), hence $C \subseteq \downarrow x$ and consequently, $m(\bot) \leqslant x$.

Luigi Santocanale (ed.): Fixed Points in Computer Science 2010, pp. 83-87

*The author is supported by grant number N206 3761 37 funded by Polish Ministry of Science and Higher Education.

2 The setup

The fundamental reason why we can treat both order and metric fixpoint theorems from a unified perspective is that both structures are examples of quantale-enriched categories [11]. Consequently, we will work with distances that do not posses all flexibility of metrics. This is of course the price that must be paid for generality of the setup.

Our primary object of investigation is thus a set X together with a distance into $[0,\infty]$ that satisfies: (a) $X(x,y) = X(y,x) = 0$ iff $x = y$, (b) $X(x,y) + X(y,z) \geqslant X(x,z)$, for all $x,y,z \in X$. In the literature on the subject a set X as above is known as a *generalized metric space* (*gms*) [2].

Any gms carries an intrinsic partial order: $x \leqslant_X y$ iff $0 = X(x,y)$. It is crucial to observe that if the distance map $X(-,-)$ takes only two values: 0 and ∞, then the gms X *is* a partial order. We will say: "a gms X is a poset" precisely when its distance map takes values in $\mathbf{2} := \{0,\infty\}$.

Taking into account absence of symmetry, every gms X has its *dual*, denoted X^{op}. Note also that $[0,\infty]$ itself is a gms with $[0,\infty](x,y) = y - x$, where $-$ is the (extended) substraction truncated at zero.

A map $f\colon X \to Y$ is *non-expansive* if $X(x,y) \geqslant Y(fx,fy)$ for all $x,y \in X$.

The set all non-expansive maps of type $X^{op} \to [0,\infty]$ will be denoted $\widehat{X}$, to distinguish it from the set $[0,\infty]^X$ of *all* maps of the same type. The former plays a crucial role in our paper: it is a gms when equipped with the usual sup-distance $\widehat{X}(\phi,\psi) := \sup_{z\in x}(\psi z - \phi z)$.

2.1 The generalized direct image of a function

Let $f\colon X \to Y$ be any non-expansive map. Consider $f^*\colon \widehat{X} \to \widehat{Y}$ given by

$$f^*(\phi)(y) := \inf_{x\in X}(\phi x + Y(y,fx)).$$

Since $\widehat{X}(\phi,\psi) + f^*(\phi)(y) \geqslant \inf_{x\in X}(\psi x + Y(y,fx)) = f^*(\psi)(y)$, we get $\widehat{X}(\phi,\psi) \geqslant \widehat{Y}(f^*(\phi),f^*(\psi))$, i.e. f^* is non-expansive. The map f^* can be interpreted as a *generalized direct image* of f, see Sect. 2.6 of [13]. For example if X and Y are posets (i.e. gmses with distances into $\mathbf{2}$), then f^* is a lower closure of the direct image of f.

It is perhaps helpful for the reader if we issue a warning here: a non-expansive map in a gms *may not be continuous*. For example, if X and Y are posets, non-expansiveness means order-preserving, while continuous means directed-sup preserving, which is a stronger property.

2.2 Completeness of gmses

Consider an operation $\mathscr{J}$ that assigns to every gms X a subset $\mathscr{J}X$ of elements of $\widehat{X}$ in such a way that (a) each $\mathscr{J}X$ contains all $\mathsf{y}x$ for all $x \in X$ (where $\mathsf{y}x := X(-,x)$); (b) each $\mathscr{J}X$ contains all $f^*(\phi)$ for $\phi \in \mathscr{J}Y$ and a non-expansive $f\colon Y \to X$. We say that elements of $\mathscr{J}X$ are *ideals on* X. Observe that $\mathscr{J}X$ with the distance inherited from $\widehat{X}$ is in fact a subgms of $\widehat{X}$.

Now we say that X *is* $\mathscr{J}$*-complete* if there exists a non-expansive map $\mathsf{S}\colon \mathscr{J}X \to X$, called *supremum*, with

$$\forall \phi \in \mathscr{J}X\ \forall x \in X\ \ X(\mathsf{S}\phi,x) = \widehat{X}(\phi,\mathsf{y}x).$$

The importance of the above notion of completeness lies in the fact that certain choices of $\mathscr{J}$ capture completeness of metric spaces and directed-completeness of posets *at the same time*, as will be shown on examples below.

3 A pattern for Banach's and Knaster-Tarski's theorems

A class $\mathscr{J}$ of ideals is *admissible* if each $\mathscr{J}$-complete X is directed-complete with respect to the intrinsic order $\leqslant_X$ introduced in Sect. 2.

Theorem 3.1. *Fix an admissible class of ideals $\mathscr{J}$. Let $T: X \to X$ be a non-expansive map on an $\mathscr{J}$-complete gms X. Suppose that there exists $\phi \in \mathscr{J}X$, which is a fixed point of T^*. Then T has a fixed point, which is the least fixed point of T above $\mathsf{S}\phi$.*

Proof. Since X is complete, $\mathsf{S}\phi$ exists. By Prop. 2.5. of [13], non-expansiveness of T yields $\mathsf{S}(T^*(\phi)) \leqslant_X T(\mathsf{S}\phi)$, hence $\mathsf{S}\phi \leqslant_X T(\mathsf{S}\phi)$. We can now follow steps of Pataraia's construction, the only difference being that instead of $\bot$, we use $\mathsf{S}\phi$. Thus T has a fixed point, which is of the form $m(\mathsf{S}\phi)$ for some $m: C \to C$. Recall that $C \subseteq X$ is the smallest set closed under T and directed lubs that contains $\mathsf{S}\phi$. Suppose now that x_1 is also fixed by T, and that $\mathsf{S}\phi \leqslant x_1$. Being the lower cone of a fixed point, $\downarrow x_1$ is closed under T and directed lubs. As a consequence $m(\mathsf{S}\phi) \in C \subseteq \downarrow x_1$. □

We will now demonstrate that Thm. 3.1 offers a pattern for generalizing and proving both of the classic fixed point theorems.

3.1 Banach's fixed point theorem

A first example of an admissible class of ideals is $\mathscr{J} = \mathbb{A}$, where by definition $\phi \in \mathbb{A}X$ if and only if $\phi x := \inf_{i\in I} \sup_{j\geq i} X(x,x_j)$ for some forward Cauchy net $(x_i)_{i\in I}$ [2] (recall that a net $(x_i)_{i\in I}$ is forward Cauchy if $\forall \varepsilon > 0\ \exists N \in I\ \forall m \geq n \geq N\ \varepsilon > X(x_n,x_m)$). It can be shown that a metric space X is $\mathbb{A}$-complete iff it is complete in the usual sense. On the other hand, if X is a poset, then X is $\mathbb{A}$-complete iff it is a dcpo.

Lemma 3.2. *$\mathbb{A}$ is admissible.*

Proof. Suppose $(x_i)_{i\in I}$ is $\leqslant_X$-directed. Define $\phi := \inf_{i\in I} \sup_{j\geq i} X(-,x_j)$. Since X is $\mathbb{A}$-complete, the supremum $\mathsf{S}\phi \in X$ exists, and now will show that it is the least upper bound of $(x_i)_{i\in I}$.

Let $k \in I$. Firstly observe $\phi(x_k) \leqslant \sup_{j\geq k} X(x_k,x_j) = 0$. Therefore $0 = \phi(x_k) = \widehat{X}(\mathsf{y}x_k, \phi) \geqslant X(x_k, \mathsf{S}\phi)$, whence $x_k \leqslant_X S\phi$. On the other hand take any upper-bound $u \in X$ of $(x_i)_{i\in I}$. Then $0 = X(x_k,u)$ for all $k \in I$, and consequently $0 = \inf_k \sup_{j\geq k} \widehat{X}(\mathsf{y}x_k, \mathsf{y}u) = \widehat{X}(\phi, \mathsf{y}u) = X(\mathsf{S}\phi, u)$, i.e. $\mathsf{S}\phi \leqslant_X u$. □

Lemma 3.3. *Let $T: X \to X$ be a non-expansive map on an $\mathbb{A}$-complete gms X. Suppose that there exists $x_0 \in X$ such that $(T^n x_0)_{n\in\omega}$ is a forward Cauchy sequence. Then there is $\phi \in \mathbb{A}X$ such that $\phi = T^*(\phi)$.*

Proof. Define an ideal on X by $\phi := \inf_{m\in\omega} \sup_{n\geq m} X(-,T^n x_0)$. Let us first show that $\phi \leqslant_{\widehat{X}} T^*\phi$. Fix $N \in \omega$, $y \in X$ and choose $\varepsilon > 0$ such that $\varepsilon > \sup_{n\geq N} X(y,T^n x_0)$. Then there is $\delta > 0$ with $\varepsilon > \sup_{n\geq N} X(y,T^n x_0) + \delta$. Use Cauchyness to get $M \geq N$ such that for all $i \geq M$, $\delta > X(T^M x_0, T^i x_0)$. Hence $\delta \geqslant \sup_{i\geq M} X(T^M x_0, T^i x_0)$. Consequently,

$$\begin{aligned}
\varepsilon > \sup_{n\geq N} X(y,T^n x_0) + \delta &\geqslant \sup_{n\geq M} X(y,T^n x_0) + \delta \\
&\geqslant X(y,T^{M+1}x_0) + \sup_{i\geq M} X(T^M x_0, T^i x_0) \\
&\geqslant \inf_{z\in X} (X(y,Tz) + \sup_{i\geq M} X(z,T^i x_0)) \\
&\geqslant \inf_{z\in X} (X(y,Tz) + \inf_{M\in\omega} \sup_{i\geq M} X(z,T^i x_0)) \\
&= T^*(\phi)(y).
\end{aligned}$$

Since ε is arbitrary, $\sup_{n\geq N} X(y,T^n x_0) \geqslant T^*(\phi)(y)$. This holds for any $N \in \omega$ and $y \in X$, therefore $\phi \leqslant T^*(\phi)$.

For the converse, let $y,z \in X$, $m \in \omega$ and $n \geq m$. Then $X(y,Tz) + X(z,T^n x_0) \geqslant X(y,Tz) + X(Tz,T^{n+1}x_0) \geqslant X(y,T^{n+1}x_0)$ and thus $X(y,Tz) + \phi z \geqslant \phi y$. Consequently, $T^*(\phi) \leqslant_{\widehat{X}} \phi$. □

Theorem 3.4 (Banach)**.** *A contraction on an $\mathbb{A}$-complete gms has a unique fixed point.*

Proof. Let X be $\mathbb{A}$-complete and $T\colon X \to X$ be contractive. Then for any $x_0 \in X$, $(T^n x_0)_{n\in\mathbb{N}}$ is a forward Cauchy sequence. Hence by Lemma 3.3, there exists $\phi \in \mathbb{A}X$ that is fixed under T^*. Now Theorem 3.1 applies and we get a fixed point of T. Since T is a contraction, any choice of x_0 leads to the same $\mathsf{S}\phi \in X$. Hence the fixed point of T is unique. □

3.2 Knaster-Tarski's fixed point theorem

Now $\mathscr{J}X = \widehat{X}$. It is well-known that that any $\widehat{(\cdot)}$-complete X is a complete lattice in the intrinsic order, and hence $\widehat{(\cdot)}$ is admissible.

It is actually a metric version of Knaster-Tarski's theorem that we are going to prove. We recover the original statement exactly when X is a poset.

Theorem 3.5 (Knaster-Tarski)**.** *A non-expansive map $T\colon X \to X$ on an $\widehat{(\cdot)}$-complete gms has the least and the greatest fixed point.*

Proof. Let $\bot, \top$ be the least and the top elements of X, respectively. By proofs of Lemmata 3.2, 3.3, $\phi z := \inf_{m\in\omega} \sup_{n\geq m} X(z,T^n(\bot))$ is an ideal on X with supremum $\bigvee_{n\in\omega} T^n(\bot)$. Moreover, ϕ is a fixed point of T^*. By Theorem 3.1, there exists a fixed point of T, which is least above $\bigvee_{n\in\omega} T^n(\bot)$, and hence least above $\bot$ as well.

Since T is non-expansive on X iff it is non-expansive on X^{op}, the same construction applied to X^{op} in place of X produces the least fixed point of T in X^{op}, i.e. the greatest fixed point of X. □

4 A pattern for Caristi's and Bourbaki-Witt's fixed point theorems

It is known from [6] that any complete metric space X can be embedded onto the set of maximal elements of a continuous dcpo $\mathbb{B}X := \{\langle x,r\rangle \mid x \in X, \infty > r \geqslant 0\} \subseteq \widehat{X}$, where $\langle x,r\rangle(z) := X(z,x) + r$, and $\langle x,r\rangle \leqslant_{\mathbb{B}X} \langle y,s\rangle$ iff $\langle x,r\rangle \leqslant_{\widehat{X}} \langle y,s\rangle$ iff $r \geqslant X(x,y) + s$. In [9] it has been shown that X is an $\mathbb{A}$-complete gms iff $(\mathbb{B}X, \leqslant_{\mathbb{B}X})$ is a dcpo. We will use this knowledge in proving a generalized version of Caristi's fixed point theorem.

Theorem 4.1. *Let X be an $\mathbb{A}$-complete gms and let $\varphi\colon X \to [0,\infty)$ be a lower semicontinuous function. Suppose $T\colon X \to X$ is an arbitrary mapping which satisfies $\varphi(x) \geqslant X(x,Tx) + \varphi(Tx)$ for each $x \in X$. Then T has a fixed point.*

Proof. Let Z be a subgms of $\mathbb{B}X$ consisting of elements of the form $\langle x,\varphi(x)\rangle$ for $x \in X$. Hence $\langle x,\varphi(x)\rangle \leqslant_Z \langle y,\varphi(y)\rangle$ iff $\varphi(x) \geqslant X(x,y) + \varphi(y)$. Moreover φ is lower semicontinuous iff it preserves directed suprema with respect to the order $\leqslant_Z$. All this means that $(Z,\leqslant_Z)$ is a subdcpo of $(\mathbb{B}X,\leqslant_{\mathbb{B}X})$. By assumption, $T'\colon Z \to Z$, defined as $T'(\langle x,\varphi(x)\rangle) := \langle Tx,\varphi(Tx)\rangle$ is expanding.

Therefore we use the Bourbaki-Witt argument (that can be found in [10]) to conclude that $T'\colon Z \to Z$ has a fixed point. That is: $\langle z,\varphi(z)\rangle = \langle Tz,\varphi(Tz)\rangle$ for some $z \in X$. Therefore, $z = Tz$, as required. □

Note that when X is a poset and φ is constant, then Thm. 4.1 becomes the Bourbaki-Witt theorem for dcpos. On the other hand, since any complete metric space is $\mathbb{A}$-complete, we have:

Theorem 4.2 (Caristi)**.** *Let X be a complete metric space and let $\varphi\colon X \to [0,\infty)$ be a lower semicontinuous function. Suppose $T\colon X \to X$ is an arbitrary mapping which satisfies $\varphi(x) \geqslant X(x,Tx) + \varphi(Tx)$ for each $x \in X$. Then T has a fixed point.*

We conclude the paper with recalling a result of A. Bauer [1] who proved that in the effective topos, there is a chain-complete lattice with an inflationary map that does not have a fixed point. As a consequence the Bourbaki-Witt theorem for chain-complete lattices (or — equivalently [4] — for dcpos) cannot have a proof in higher-order intuitionistic logic. For us, this means that Theorem 4.1 has no constructive proof. Moreover, if the original Caristi theorem for metric spaces had a constructive proof that does not rely on symmetry of the distance, nor on the assumption that $X(x,y) < \infty$ for all $x, y \in X$ — see e.g. [8], [5] for proofs that neither use Zorn's lemma nor symmetry — then it would be a valid proof of our Theorem 4.1, and hence of the Bourbaki-Witt construction, which is impossible.

Taking into account the observations above we conjecture that symmetry plays no role in the argument, and therefore *no proof* of Caristi's theorem is constructively valid, but we are unable to prove this at the moment. One way to support our claim would be a construction, for every complete metric space X, of a non-symmetric $\mathbb{A}$-complete quasi-metric on X with equivalent fixed-point properties; such a quasi-metric could perhaps be found by a deeper analysis of these existing proofs of Caristi's theorem that do not invoke equivalents of Zorn's lemma.

References

[1] Bauer, A. (2009) On the Failure of Fixed Point Theorems for Chain-complete Lattices in the Effective Topos, *Electronic Notes in Theoretical Computer Science* 249, p. 157–167.

[2] Bonsangue, M., van Breugel, F., Rutten, J. (1998) Generalized metric spaces: completion, topology, and powerdomains via the Yoneda embedding. *Theoretical Computer Science* 193, p. 1–51.

[3] Caristi, J. (1976) Fixed point theorems for mapping satisfying inwardness conditions, *Transactions of the American Mathematical Society* 215, p. 241–251.

[4] Dacar, F. (2008) Suprema of families of closure operators, Seminar for foundations of mathematics and theoretical computer science, Faculty of Mathematics and Physics, University of Ljubljana, Slovenia. Available on-line at: http://dis.ijs.si/France//

[5] Dugundji, J. and Granas, A. (1982) Fixed point theory. Vol. 61 of Monografie Matematyczne, Polish Scientific Publishers (PWN), Warszawa.

[6] Edalat, A. and Heckmann, R. (1998) A computational model for metric spaces. *Theoretical Computer Science* 193, p. 53–73.

[7] Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M. and Scott, D.S. (2003) Continuous lattices and domains. Volume 93 of *Encyclopedia of mathematics and its applications*.

[8] Goebel, K., Kirk, W.A. (1990) Topics in metric fixed point theory. Vol. 28 of Cambridge Studies in Advanced Mathematics. Cambridge University Press.

[9] Kostanek, M. and Waszkiewicz, P. The formal ball model for Q-categories. Submitted.

[10] Lang, S. (2002) Algebra. Revised Third Edition. Appendix 2. Springer-Verlag, New York.

[11] Lawvere, F.W. (1973) Metric spaces, generalized logic, and closed categories. *Rend. Sem. Mat. Fis. Milano* 43, pp. 135–166. Also in: Repr. Theory Appl. Categ. 1:1–37, 2002.

[12] Pataraia, D. (1997) A constructive proof of Tarski's fixed-point theorem for dcpos. 65th Peripatetic Seminar on Sheaves and Logic, November 1997.

[13] Waszkiewicz, P. (2009) On domain theory over Girard quantales. *Fundamenta Informaticae* 92, p. 1–24.

Author Index